Facultad de Informática
Universidad Complutense de Madrid

Implementación de un planificador de tareas para multitarea hardware sobre un procesador empotrado

Memoria de Proyecto de Sistemas Informáticos

Isabel Fernández Sánchez
Ángel Morales Rodríguez
Marta Salgado Rivera

Profesor Director: Sara Román

2006/2007

Índice

1. Autorización.……..4

2. Resumen……..5

3. Objetivo……...6

4. Introducción……7

FPGA Virtex-II Pro.…….…..8

Reconfiguración Parcial Dinámica…….……..11

PowerPC…….……13

EDK…….……14

Xilinx Platform Studio (XPS)

Impact

XMD

5. Metodología…….16

6. Descripción del Algoritmo Implementado……29

7. Resultados y Conclusiones…….38

- **Ejemplo 1**
- **Ejemplo 2**
- **Ejemplo 3**
- **Simulación del algoritmo implementado**

8. Conclusiones Generales.…….53

9. Bibliografía……..54

Índice de Figuras Relevantes

1. **Estructura de una FPGA……..…..8**

2. **Características de distintos tipos de FPGAs……..…...9**

3. **Familias de FPGAs……..……9**

4. **Virtex-II/Virtex-II Pro…….……..10**

5. **CLBs de Virtex-II y Virtex-II Pro……10**

6. **FPGA reconfigurable en 1D……..11**

7. **FPGA reconfigurable en 2D…….….12**

8. **Conexión al bus OPB de periféricos del usuario…....15**

1. Autorización

Los autores de este proyecto, Isabel Fernández Sánchez, Angel Morales Rodríguez y Marta Salgado Rivera, autorizan mediante este texto a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos no comerciales, y mencionados expresamente a sus autores tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Fdo: Isabel Fernández Sánchez	Fdo: Ángel Morales Rodríguez	Fdo: Marta Salgado Rivera

2. Resumen

Resumen

El objetivo del proyecto es programar un planificador de tareas sobre un procesador empotrado en una FPGA *(field programmable gate array),* dispositivo semiconductor que contiene componentes lógicos programables e interconexiones programables entre ellos, que gestione la carga de tareas hardware sobre dicho dispositivo de acuerdo con un algoritmo basado en la división de la FPGA en particiones de distinto tamaño.

Este algoritmo consistirá en asignar la partición en la que se va a ejecutar cada tarea, antes de que la tarea llegue al procesador. El lenguaje de programación utilizado para programar este algoritmo es C.

La implementación se realizará en una placa de prototipado que dispone de una FPGA Virtex II Pro.

Palabras clave

FPGA, EDK, planificación de tareas, partición, reconfiguración parcial, CLB

Abstract

The goal of our Project is to programme HW task schedular in a embedded processor in a FPGA (*(field programmable gate array),* semiconductor device which contains logical programming components and programming interconnections between them, to administer the hardware tasks load in this device according to an algorithm based on the FPGA's partition where every partition will have different size.

This algorithm chooses the partition where every task is going to be executed before being processed by the embedded processor. The lenguage of programming that we used is C.

The implementation will be carried out in an environment that contains a FPGA Virtex II Pro.

Key words

FPGA, EDK, tasks scheduling, partition, partial reconfiguration, CLB

3. Objetivo

El objetivo del proyecto es la implementación de un planificador de tareas para multitarea hardware sobre un procesador PowerPC en una FPGA Virtex-II Pro.
La reconfiguración de la FPGA se realizará en una dimensión y se considerará dicha FPGA dividida en cuatro particiones de tamaños fijos.

4. Introducción

Una FPGA es un dispositivo semiconductor que incluye un conjunto programable de bloques combinacionales o de almacenamiento (CLBs), bloques de memoria y una red programable de interconexión entre ellos.

En el presente proyecto se utilizará un modelo concreto de FPGA de la familia de FPGAs de Xilinx, la Virtex-II Pro.

Las particularidades de este modelo de FPGA son varias:

- Incluye dos microprocesadores PowerPC 405 empotrados en la placa, así como diferentes periféricos.
- Soporta la reconfiguración parcial dinámica del sistema. Se permite reprogramar una parte del diseño sin que afecte a la función del resto. Se considera por tanto un "sistema reconfigurable dinámicamente".
- La reconfiguración dinámica se hace cargando un bitstream que afecta únicamente a la parte de la placa que se quiere configurar. Este fichero de bitstream incluye todos los datos y comandos necesarios para la configuración.
- La reconfiguración de la FPGA se realiza en una única dimensión, dejando de lado la reconfiguración en dos dimensiones, más compleja y costosa.

La planificación de la reconfiguración y ejecución de las tareas en la placa serán llevadas a cabo por un algoritmo planificador de tareas. Dicho algoritmo se centrará en el mantenimiento de la información de las tareas a ejecutar, el control de admisión/planificación de dichas tareas y su ubicación en una zona específica de la placa FPGA.

La implementación del algoritmo se realiza sobre el Embedded Development Kit (EDK) de Xilinx que permite introducir código C.
La herramienta EDK permite además especificar qué dispositivos se desean añadir a la placa. Estos dispositivos actúan junto con el algoritmo para generar un bitstream o .bit que se descarga sobre la placa.

La descarga de dicho .bit sobre la placa se realiza a través de la herramienta Impact de Xilinx que permite programar la FPGA.

Para finalizar, el EDK nos ofrece la herramienta Xilinx Microprocessor Debugger (XMD) encargada de descargar el código de la aplicación, verificar su funcionamiento e introducir el código en la memoria DDR.

4.1 FPGA Virtex-II Pro

Una **FPGA**(*field programmable gate array)* es un dispositivo semiconductor que incluye bloques lógicos combinacionales (CLBs), elementos de memoria y una red de interconexión programable.

Las FPGAs pertenecen al conjunto de dispositivos programables que introdujo Xilinx en el mercado.

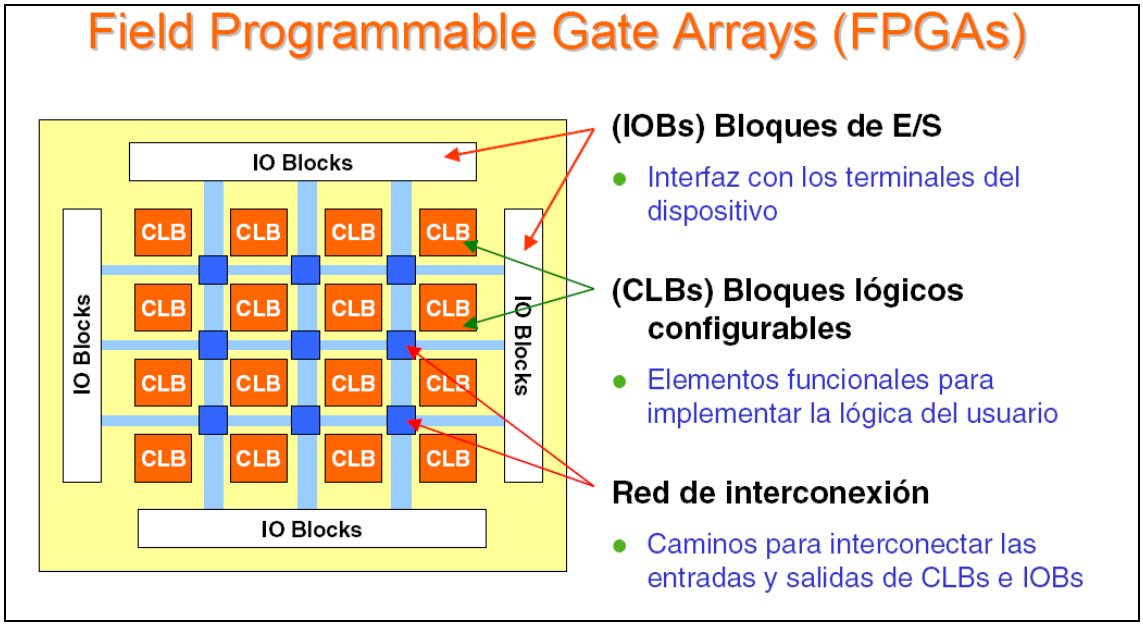


Figura1: Estructura de una FPGA

Hoy en día existen diversidad de FPGAs, atendiendo a diversos criterios:

- Según su Arquitectura, podemos encontrar los gate arrays (filas), matriz simétrica y mar de puertas.
- Según su Tipo de Bloque Lógico tenemos: puertas lógicas, multiplexores y tablas de búsquedas (LUT).
- Según la Interconexión, están los canales de routing y red de interconexión.
- Y las técnicas de Programación pueden ser antifusibles, de memoria no volátil y de memoria RAM.

A continuación se muestran algunos de los tipos de FPGAs existentes, detallando algunas de sus principales características:

Compañía	Arquitectura	Bloques lógicos	Tecnología de programación
Actel	Gate array	Multiplexores	fusibles
AMD	PLDs jerárquicos	PLDs	EEPROM
Altera	PLDs jerárquicos	PLDs	EPROM
Plessey	Mar de puertas	Puertas NAND	RAM
Xilinx	Matriz simétrica	Tabla look-up	RAM

Figura2: Características de distintos tipos de FPGAS

Virtex incluye a su vez diferentes familias de FPGAs:

Familia	Características	Diseño digital	Procesadores empotrados	Arquitectura computadores	DSP	SOC
XC4000	1.6K – 85K puertas	✓				
Spartan	5K – 40K puertas	✓				
Spartan-II	15K – 300K puertas BRAM: 16K – 64K	✓	✓	✓		
Virtex	58K – 1.2M puertas BRAM: 32K – 132K	✓	✓	✓		
Virtex-II	40K – 8M puertas BRAM: 72K – 3M 4 – 168 multiplicadores		✓	✓	✓	
Virtex-II Pro	Virtex-II + 0 – 4 PPC 4 – 24 Rocket I/O transc. 12 – 556 multiplicadores			✓	✓	✓

Figura3: Familias de FPGAs

En el presente proyecto se realiza el estudio sobre una de las familias de FPGAs de Xilinx, la Virtex-II Pro.

La **Virtex-II Pro** es una FPGA que además de los bloques lógicos e interconexiones programables que caracterizan a toda FPGA, tiene también microprocesadores y periféricos.

Este modelo de FPGA, incluye hasta dos procesadores PowerPC 405, núcleos de procesadores Risc, de 32-bits en un solo dispositivo, empotrados junto a la lógica de la FPGA. El PowerPc 405 de IBM está integrado en el dispositivo Virtex-II Pro usando la

arquitectura IP-immersion, la cual permite la comunicación en el interior de toda la estructura de la FPGA. A continuación se muestran algunas de las características de las FPGAs Virtex:

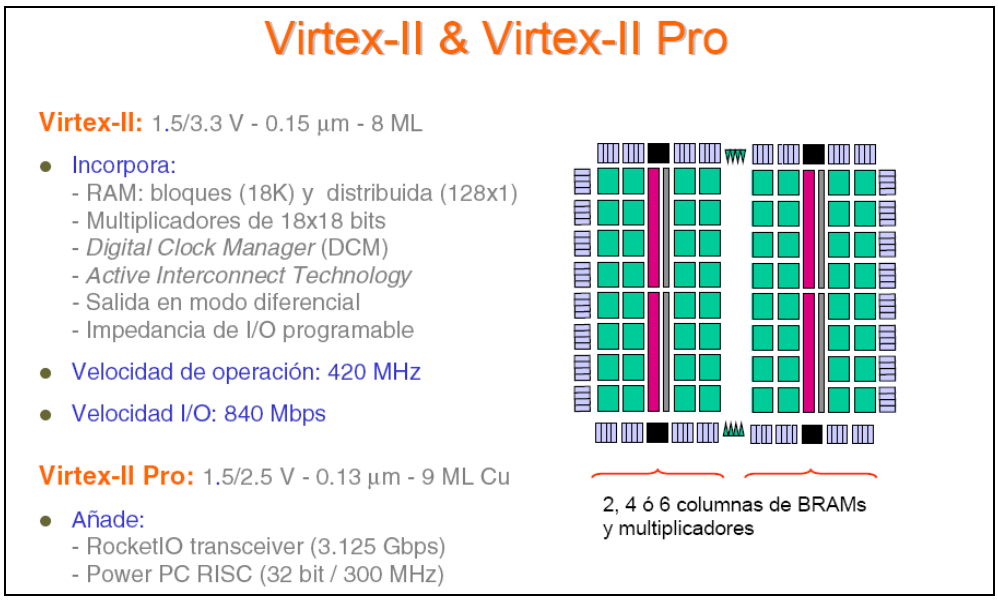


Figura4: Virtex-II/ Virtex-II Pro

El modelo Virtex soporta la reconfiguración parcial dinámica del sistema.
Esto tiene lugar cuando se produce un cambio en la configuración de una sección de la FPGA en tiempo de ejecución.
Por tanto, se considera un "sistema reconfigurable dinámicamente".

Las ventajas de usar FPGAs son la posible reprogramación de éstas una vez que han salido a la venta, pudiendo así corregir posibles errores, con lo que reduce costes de investigación, diseño y pruebas de nuevos productos. Otra alternativa serían los Dispositivos de lógica programable compleja (CPLD).
Los IOBs pueden ser usados como entrada o como salida.

En la siguiente imagen puede verse con detalle un CLB:

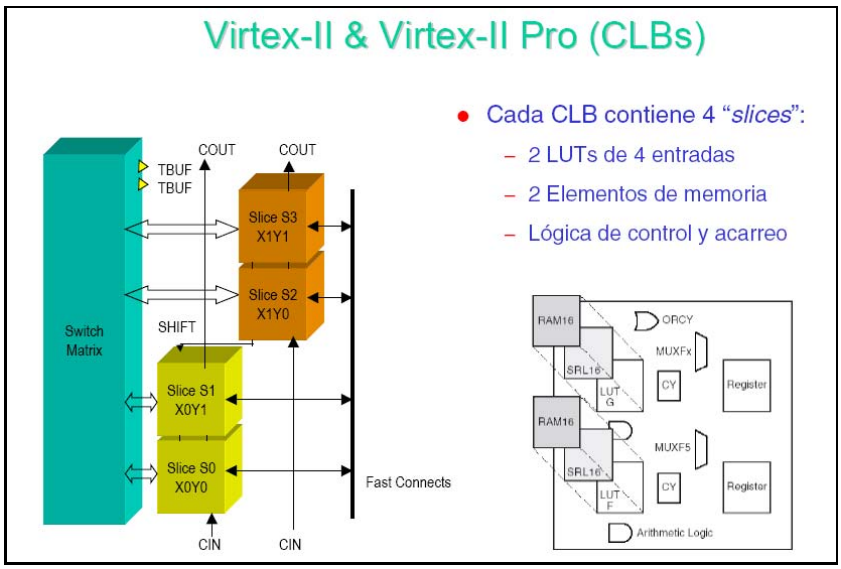


Figura5: CLBs de Virtex-II y Virtex-II Pro

4.2 Reconfiguración Parcial Dinámica

La Virtex II se reconfigura en una sola dimensión, esto quiere decir que para configurar o reconfigurar un bloque básico hay que escribir toda la columna o toda la fila de bloques básicos.

Estas FPGAs tienen una organización interna en 2D pero están diseñadas para ser reconfiguradas en una sola dimensión, por tanto, lo mínimo que se podría reconfigurar sería una columna vertical completa. Actualmente, hay herramientas que permiten la reconfiguración 2D, como podría ser el JRTR, que permitiría un mejor uso del espacio de la FPGA, si no fuera por la restricciones que dicha herramienta impone.

En el modelo de reconfiguración en una sola dimensión (familia Xilinx) la reconfiguración se hace por filas o columnas de CLBs completas.

Una de las principales innovaciones de las FPGAs Virtex, es que son capaces de realizar **reconfiguración parcial dinámica** del dispositivo, que supone una opción interesante para el desarrollo de sistemas con hardware reconfigurable. Para ello se necesita un bitstream, ya que es con lo que se configura esta FPGA, y cuando se quiere modificar algo de la configuración, se carga la parte del bitstream modificada, ya que en este fichero es donde están almacenados todos los datos y comandos de configuración.

Podría por tanto verse la FPGA como una gran superficie de procesamiento, de modo que cada tarea compilada debe asignarse a una parte disponible de la FPGA. Debe tenerse en cuenta que una tarea puede cargarse o abandonar la FPGA sin afectar al resto de tareas, del mismo modo que ocurriría en un sistema SW multitarea.

Esta capacidad de reconfiguración parcial dinámica junto con la ampliación de tamaño de dispositivos que existen hace que aparezca la posibilidad de utilizar estos dispositivos en contextos en los que antes solo se podía utilizar procesadores de propósito general.

El tamaño de las FPGAs está creciendo, lo que obliga a la aparición de un nuevo modelo de recursos que permita la ejecución HW multitarea en una FPGA que sea reconfigurable parcialmente en tiempo de ejecución.

Hasta hace poco tiempo, Los modelos de FPGA que existían se reconfiguraban en **1D** (una dimensión).

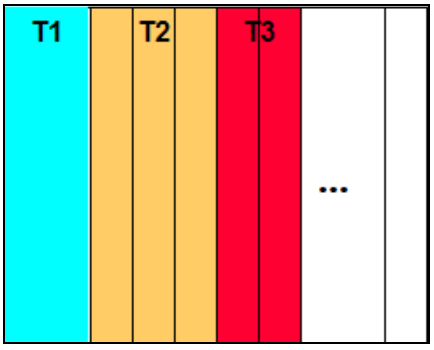


Figura6: FPGA reconfigurable en 1D

Para modelos reconfigurables dinámicamente en **2D** (dos dimensiones), el proceso de asignación de recursos resulta mucho más complejo e interesante. En este modelo, partiendo de un bloque básico dinámicamente reconfigurable bidimensional (BBDR), el HW a gestionar puede verse como un teselado regular, sobre el que pueden ubicarse las tareas HW. La reconfiguración se hace por bloques de CLBs, a discreción del usuario, como muestra la figura:

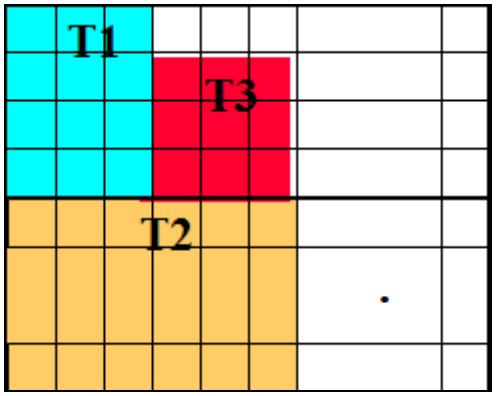


Figura7: FPGA reconfigurable en 2D

Xilinx proporciona herramientas apropiadas y máscaras, que permiten reconfigurar dinámicamente secciones de las FPGAs Virtex en 2D, y permiten por tanto hacer un uso más eficiente del espacio disponible en la FPGA.

En el presente proyecto se opta por la reconfiguración en 1D ya que la placa Virtex II Pro así lo soporta.

Dentro de la Reconfiguración Parcial Dinámica se distingue:

1. Por un lado se encuentran los modelos de **gestión complejos**, es decir, modelos de gestión de la FPGA , que asumen que se puede ubicar una tarea en cualquier punto de la FPGA, y por tanto, se requieren estructuras de datos complejas y gestión complicada del espacio libre.

2. Por otro lado, el modelo de **gestión sencilla**, en el que las áreas en las que se pueden ejecutar tareas son fijas, simplificando así la gestión del espacio libre.

El objetivo del presente proyecto es implementar un algoritmo de respuesta rápida, fácil de implementar y que resuelva el problema de ubicación de tareas *on-line* de manera ágil y eficiente en un entorno altamente dinámico, sin producir fragmentación.
Se ha enfocado este trabajo para que haga uso de las ventajas principales de un algoritmo de ubicación directo, que no da lugar a los overheads de ubicación y fragmentación con que se encuentran los algoritmos de gestión compleja y que proporciona una mayor flexibilidad a la hora de gestionar el área de la FPGA.

Se utilizará por tanto un modelo de reconfiguración en 1D que considera la FPGA dividida en 4 particiones de diferentes tamaños.

La multitarea quedará por tanto dividida en:

- ejecución SW (que a su vez podría ser multihilo)
- ejecución HW (se podrían ejecutar a la vez tantas tareas como particiones en las que está dividida la FPGA, 4 para el presente proyecto)

4.3 PowerPC (ppc)

Es un procesador de tercera generación de arquitectura de computadoras de tipo RISC creada por la Alianza AIM, un consorcio de empresas compuesto por Apple, IBM y Motorola, de cuyas primeras letras, surgió la sigla. Los procesadores de esta familia son producidos por IBM y Freescale Semiconductor que es la división de semiconductores y microprocesadores de Motorola, siendo utilizados principalmente en ordenadores o computadores Macintosh de Apple Computer.

Este microprocesador está diseñado con base en la arquitectura POWER de IBM con algunos componentes tomados del microprocesador Motorola 68000 para darle compatibilidad con arquitectura de los ordenadores de Apple. En ella pueden ser ejecutados, al menos, los sistemas operativos:

- AmigaOS/MorphOS
- BeOS
- FreeBSD
- GNU/Linux
- Mac OS
- Mac OS X
- QNX
- VxWorks
- Windows NT 3.51

El procesador PowerPC se dijo que tenía algunos problemas de temperatura, pero aún así, mantiene un amplio uso en sistemas SoC (System On Chip) y sistemas empotrados en general, como por ejemplo los Mars Rovers de la misión Mars Exploration Rover de la NASA, en la cual se utilizó una versión adaptada a la radiación del procesador Power PC para el cerebro de los vehículos de de exploración.

4.4 EDK

La implementación del algoritmo de planificación de tareas sobre el PowerPC se ha realizado con el **Embedded Development Kit** (EDK) de Xilinx.

Se trata de una herramienta que permite introducir todos los dispositivos que se desean añadir al diseño, que junto con el código C generará un .bit, que posteriormente el IMPACT descargará sobre la FPGA.

Este entorno engloba un conjunto de componentes IP(componentes de propiedad intelectual) y herramientas de diseño que facilitan el desarrollo de sistemas de procesadores empotrados sobre FPGAs de Xilinx.

4.4.1 Xilinx Platform Studio (XPS)

Es una plataforma de desarrollo que permite utilizar el lenguaje de programación C, junto con otras herramientas como el EDK, el Impact o el XMD. XPS actúa como interfaz gráfica del EDK (Embedded Development Kit).

La interfaz gráfica de usuario XPS (*Xilinx Platform Studio*) proporciona, una serie de plantillas que facilitan el desarrollo de periféricos conectables al bus OPB y PLB. Estas plantillas consisten en código VHDL que incluye dos componentes (Figura1): IPIF (*Intellectual-Property Interface*), que realiza las funciones de interfaz con el bus OPB; y User_logic, que contiene la lógica desarrollada por el usuario. En el presente proyecto, este último fichero incluirá la descripción VHDL del controlador difuso proporcionada por *Xfuzzy* más el código necesario para acceder al controlador a través de los registros del módulo-IP. Ambos componentes se comunican a través de la interfaz IPIC (*Intellectual-Property Interconnect*), que es independiente del bus periférico.

Existen diferentes tipos de plantillas dependiendo del modo de operación del periférico (maestro/esclavo) y los servicios proporcionados por el bloque IPIF. La interfaz gráfica de usuario XPS también facilita la generación de los archivos ".mdp" (*microprocessor peripheral definition*) y ".pao" (*peripheral analyze order*).
Estos ficheros son necesarios para el uso del módulo-IP en XPS como otros periféricos del sistema.

El ciclo de diseño de un controlador difuso como módulo-IP incluye los siguientes pasos:

- En primer lugar, mediante el asistente para creación de periféricos de XPS se indica el nombre y la versión del periférico, así como su lugar de destino. Seguidamente se selecciona el tipo de bus al que irá conectado.

- A continuación se configuran los diferentes servicios implementados por el bloque IPIF (operación como maestro o esclavo, soporte de interrupciones, número de registros, acceso directo a memoria, etc.).

- Por último tiene lugar la implementación de la lógica de usuario. Finalmente, el último paso es la importación del periférico. El asistente para importar periféricos de XPS requiere la inclusión de las plantillas VHDL y los elementos de librería, así como la asignación de puertos y la definición de parámetros. Una vez completados estos pasos el módulo-IP puede utilizarse como cualquier otro periférico en EDK.

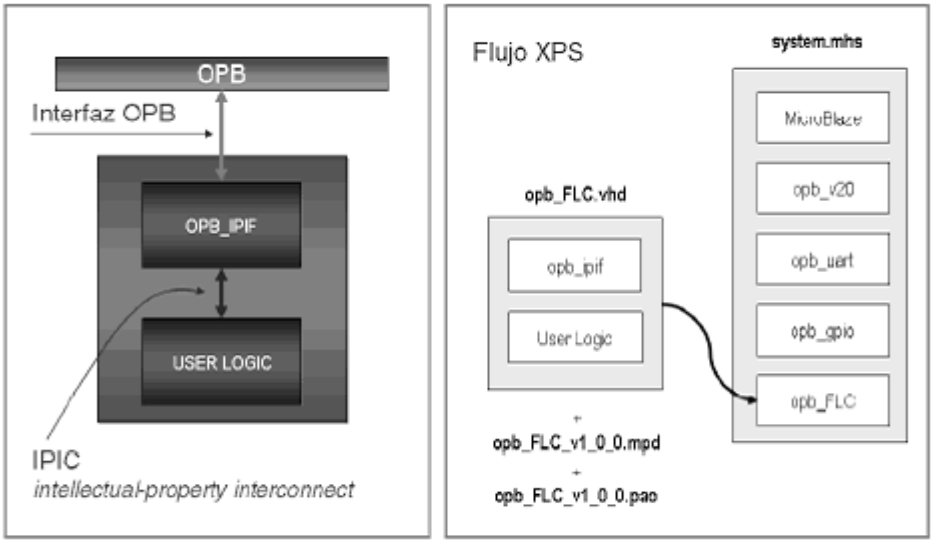


Figura8. Conexión al bus OPB de periféricos del usuario

4.4.2 Impact

Impact es una herramienta de Xilinx que permite descargar el bit-stream generado con el EDK en la FPGA. Es un accesorio de Xilinx Platform Studio, que tiene numerosos bugs y rarezas.

4.4.3 XMD

XMD (Xilinx Microprocessor Debugger) es el encargado de descargar el código de la aplicación y establecer los puntos de ruptura, así como las trazas necesarias para la verificación completa de su funcionamiento.
Es una herramienta del EDK para introducir el código del algoritmo en la DDR.

5. Metodología

5.1 Pasos o Etapas en el desarrollo del Proyecto

El proyecto podría dividirse en tres etapas:

- Primera Etapa

  Consiste en adquirir conocimientos sobre el tema general a tratar: importancia de las FPGAs y la reconfiguración parcial, y otros conceptos relevantes.
  Por otra parte es importante la toma de contacto y dominio del entorno y herramientas necesarias para el desarrollo de los dispositivos hardware que queremos incluir en el diseño, así como para la generación, compilación y ejecución de programas en la FPGA.
  El entorno y herrmamientas utilizadas comprenden:

  - El Embedded Development KIT (en adelante EDK) que es el corazón del entorno que contiene las librerías, linkers y compiladores necesarios para generar hardware y software ejecutable en la FPGA.

  - El Xilinx Platform Studio (en adelante XPS) que es la interfaz gráfica del EDK y que hace más amigable el uso del entorno por medio de ventanas.

  - El Impact como utilidad para cagar los bitstreams (archivos con extensión .bit)generados a través del XPS en la FPGA

  - El Xilinx Microprocessor Debugger para cargar archivos ejecutables (executable.elf) en la memoria externa DDRam y ejecutarlos.

  - El Hyperterminal como medio de conexión entre la FPGA y el PC a través del puerto serie para poder visualizar la traza y los mensajes de nuestro programa en la pantalla del PC

- Segunda Etapa

  La segunda etapa del proyecto consiste en la implementación de un algoritmo de planificación de tareas que decida cómo se llevará a cabo la ejecución eficiente de dichas tareas en una FPGA, considerándose esta placa dividida en cuatro particiones independientes de diferentes tamaños.
  Dicho algoritmo es implementado en lenguaje C , teniendo presente que solo son válidas las funciones de C que hayan sido compiladas con las librerías de Xilinx para la FPGA tratada y para el microprocesador empotrado en la misma.

- Tercera Etapa

  Para terminar se ejecutarán diferentes lotes de tareas que permitirán sacar conclusiones sobre la eficiencia del algoritmo utilizado y de la multitarea HW.

5.2 Procedimiento seguido

Los pasos a seguir se citan a continuación:

Lo primero que se hace es seleccionar los dispositivos disponibles de la FPGA que queremos utilizar para generar posteriormente un archivo .bit que contendrá dicho diseño.En nuestro caso como los más importantes son el microprocesador PowerPC, los LEDS, la memoria externa DDR, el puerto serie, etc.

Una vez creados los dispositivos hardware se aprenden a utilizar a través de sus funciones y librerías cada uno por separado.

Las compilaciones de estos programas de prueba, generan tres archivos principales:

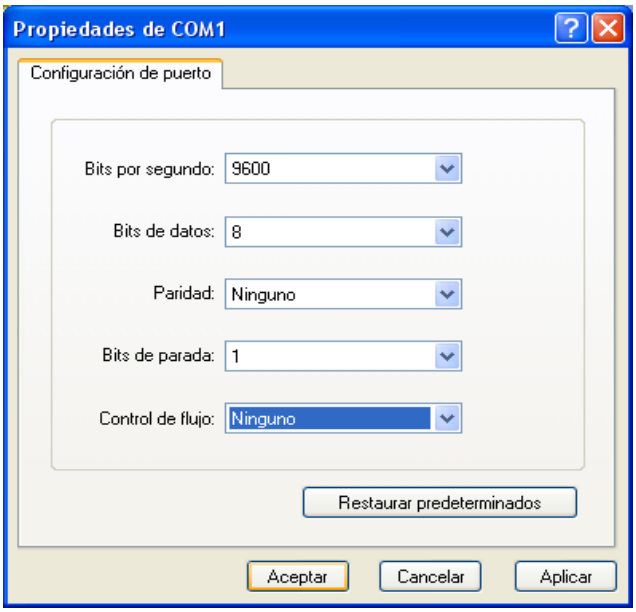
- System.bit → Contiene el bitstream con el diseño hardware de nuestra FPGA.
- Executable.elf → Contiene el programa ejecutable en lenguaje máquina.
- Download.bit → Contiene los dos elementos anteriores y toda la lógica de control necesaria para ejecutarse.

Una vez compilado hay varias formas de ejecutarlo, en este proyecto se han utilizado tres estrategias diferentes:

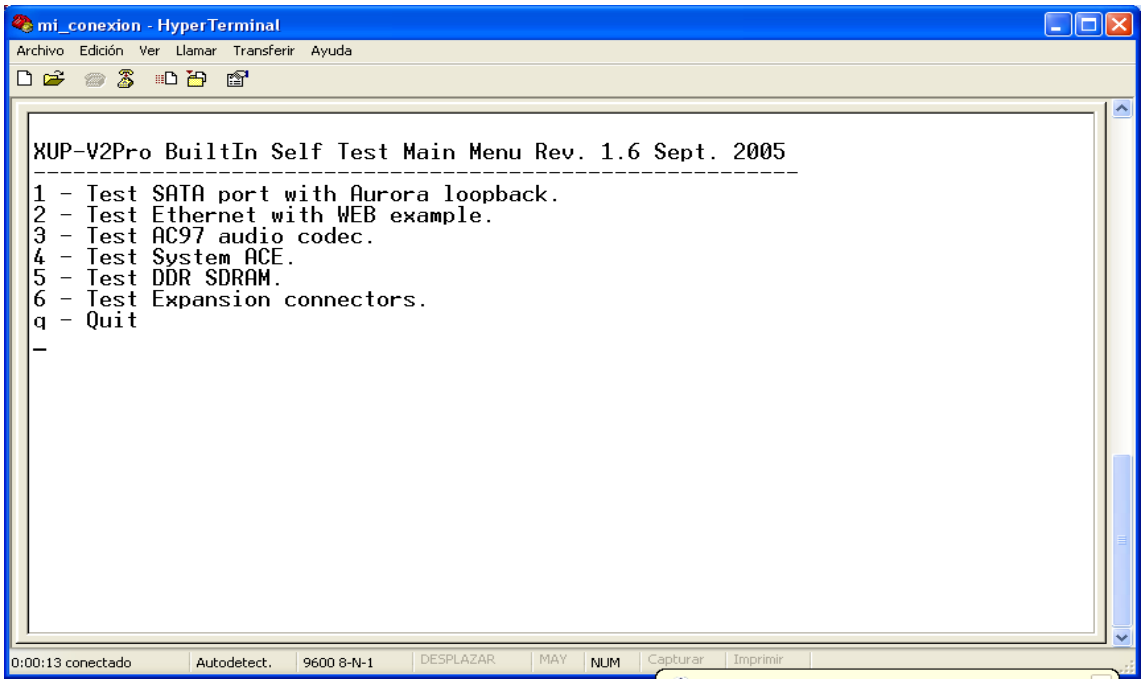
- La primera consiste en cargar por medio del Impact el archivo download.bit en la FPGA, y ejecutarlo directamente. Este proceso es viable siempre y cuando el programa ejecutable quepa en la memoria interna de la FPGA.
- La segunda consiste en cargar por medio del Impact el archivo system.bit en la FPGA y a través del XMD cargar en archivo executable.elf en la memoria externa DDR y ejecutarlo.
- La tercera consiste en cargar el proyecto a través de una memoria Flash. Esta opción permite mucha más flexibilidad. Para este fin se genera un archivo system.ace que será tratado por la FPGA y que debe contener el bitstream con el diseño hardware utilizado, el archivo executable.elf , y otra información si fuera necesario.

A continuación, se tratan en profundidad los pasos seguidos con el asistente:

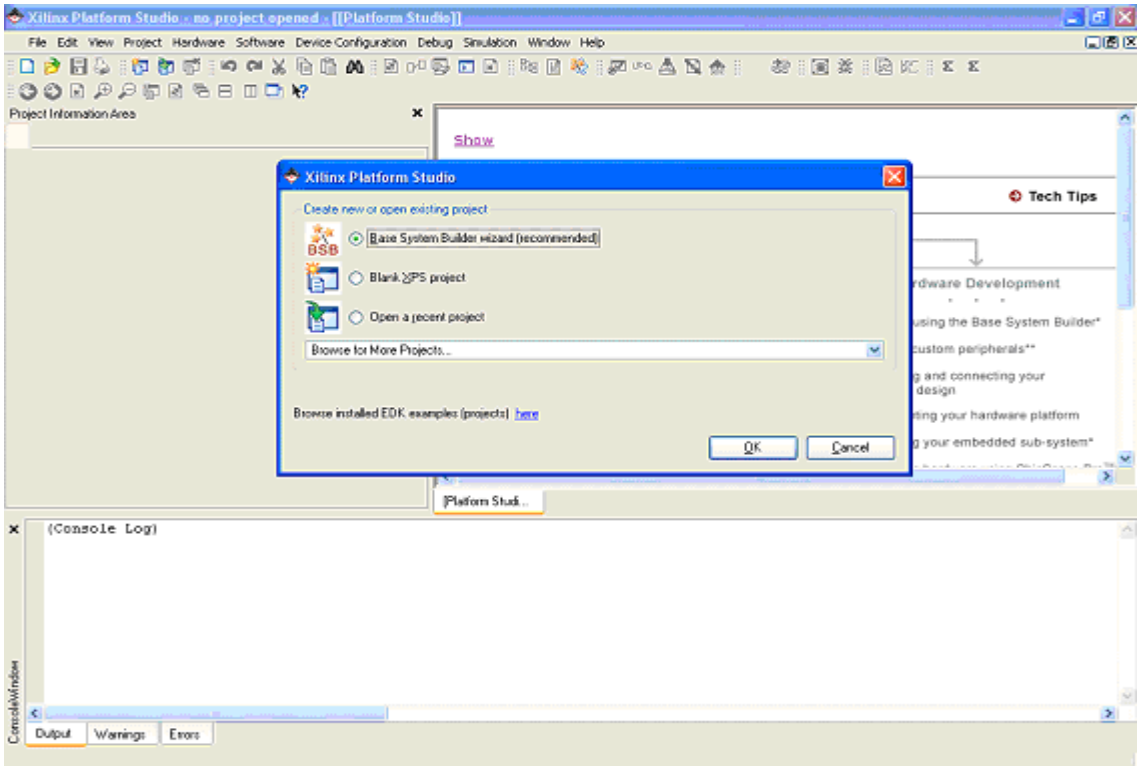
- Lo primero que se hace es encender el “hyperterminal”, indicando el nombre de la conexión y con la configuración siguiente:



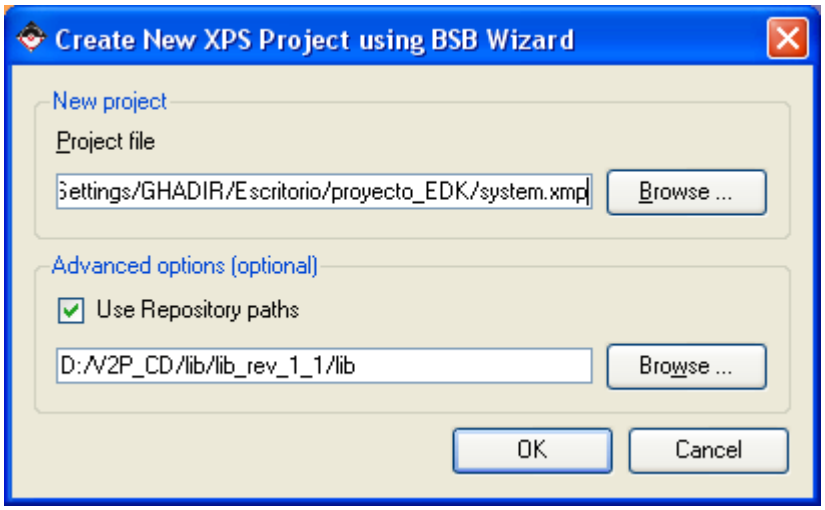
El hyperterminal permite realizar envío y recepción de mensajes, comunicándose la FPGA con el PC a través del puerto serie. De esta manera, será posible mostrar en pantalla valores de la FPGA.
Al encender la FPGA manteniéndose la configuración por defecto llamada Golden se muestra en el hyperterminal el siguiente menú por defecto que permite comprobar el buen estado de la conexión:



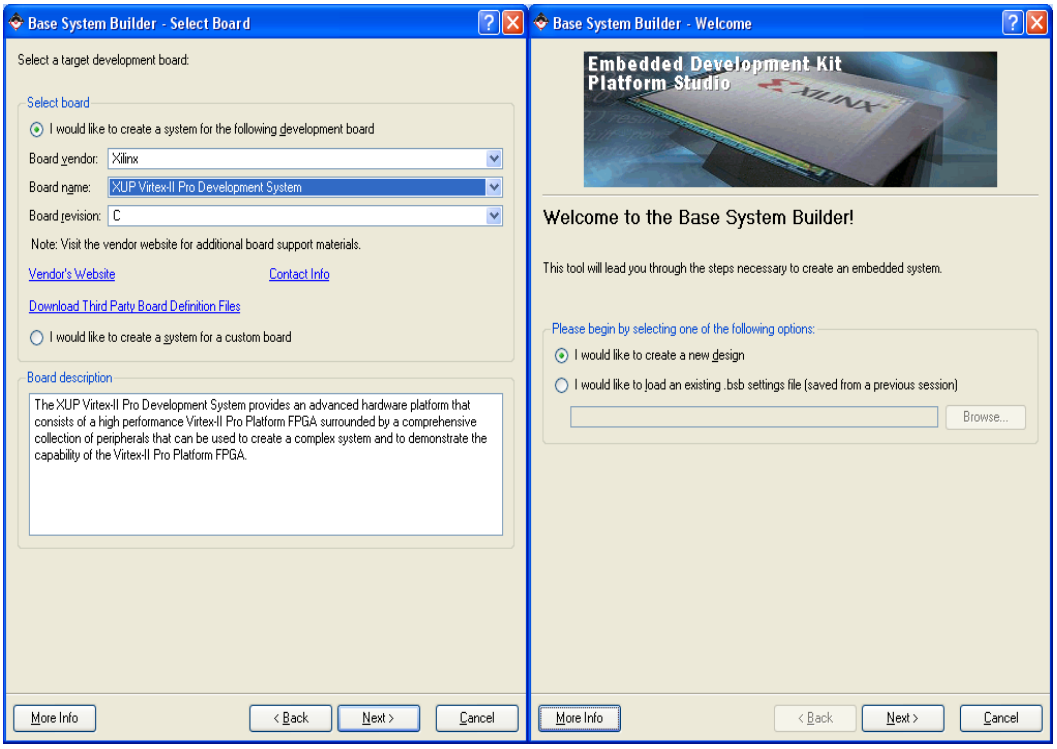
- En segundo lugar se inicia el XPS(Xilinx Platform Studio), que permite seleccionar los dispositivos hardware que se desean incluir en el proyecto:



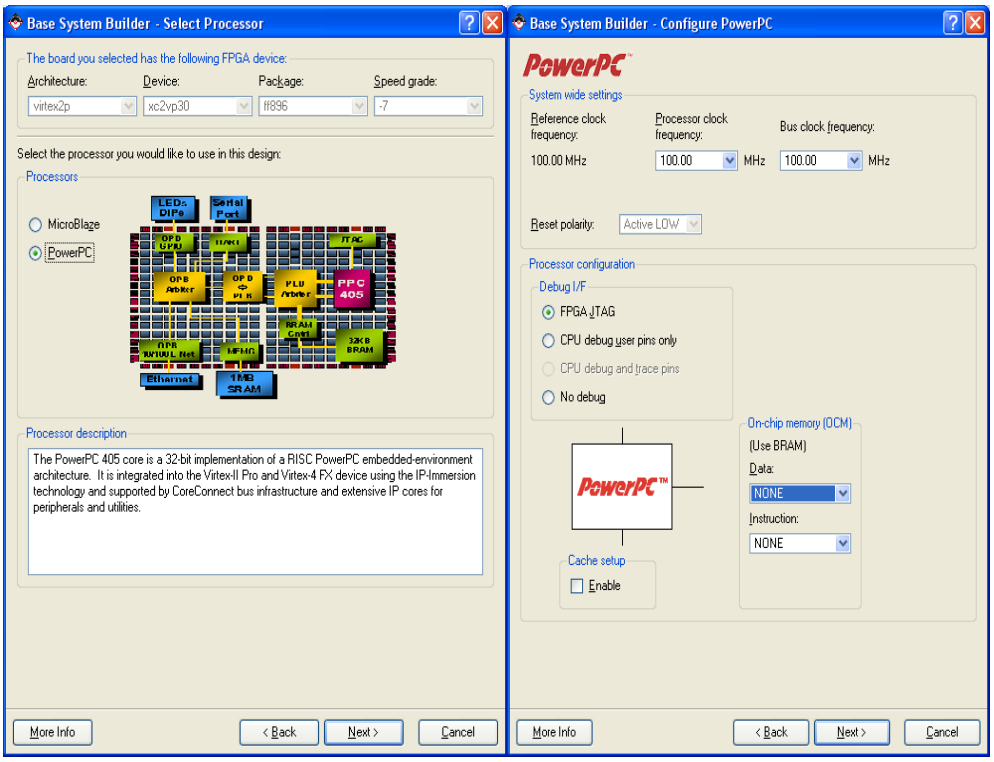
- La creación del nuevo proyecto se realiza incluyendo las librerías necesarias que permitirán utilizar el modelo de FPGA disponible en el laboratorio:



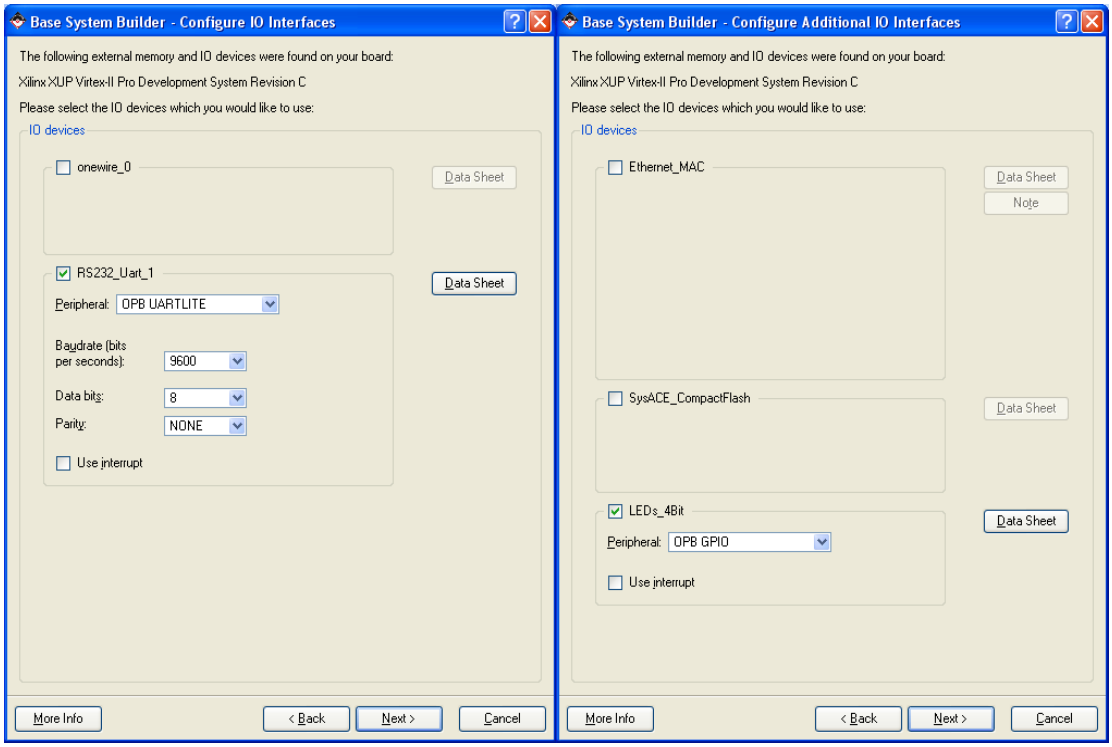
- A continuación se selecciona como placa de diseño la Virtex II que es sobre la que trabajaremos:



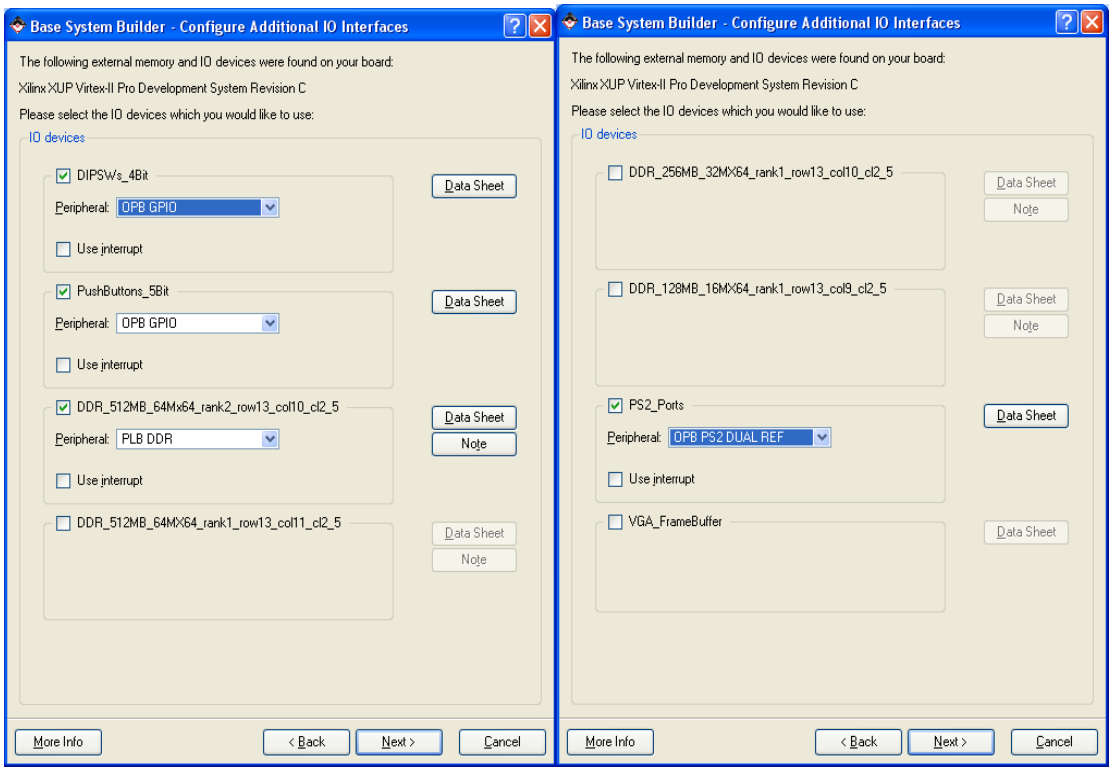
- El siguiente paso consiste en seleccionar los elementos HW que se desean utilizar en la FPGA. Para nuestro trabajo se emplea un microprocesador de tipo PowerPC. Así mismo se configura el microprocesador para que utilice el estándar de depuración y programación JTAG que nos permitirá hacerlo a través del puerto paralelo de nuestro ordenador.



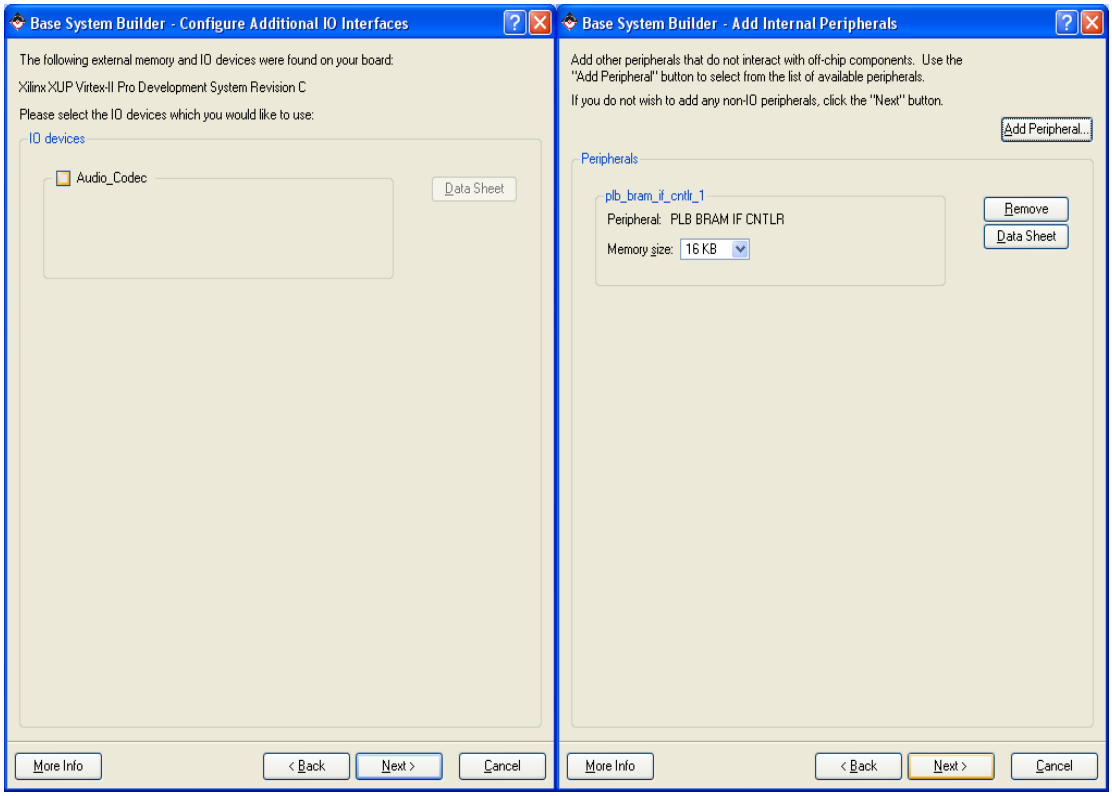
- Seleccionamos RS232_Uart_1 y los LEDS de la placa. La UART es un dispositivo emisor/receptor utilizado para realizar comunicaciones asíncronas serie de datos entre dos o más dispositivos.



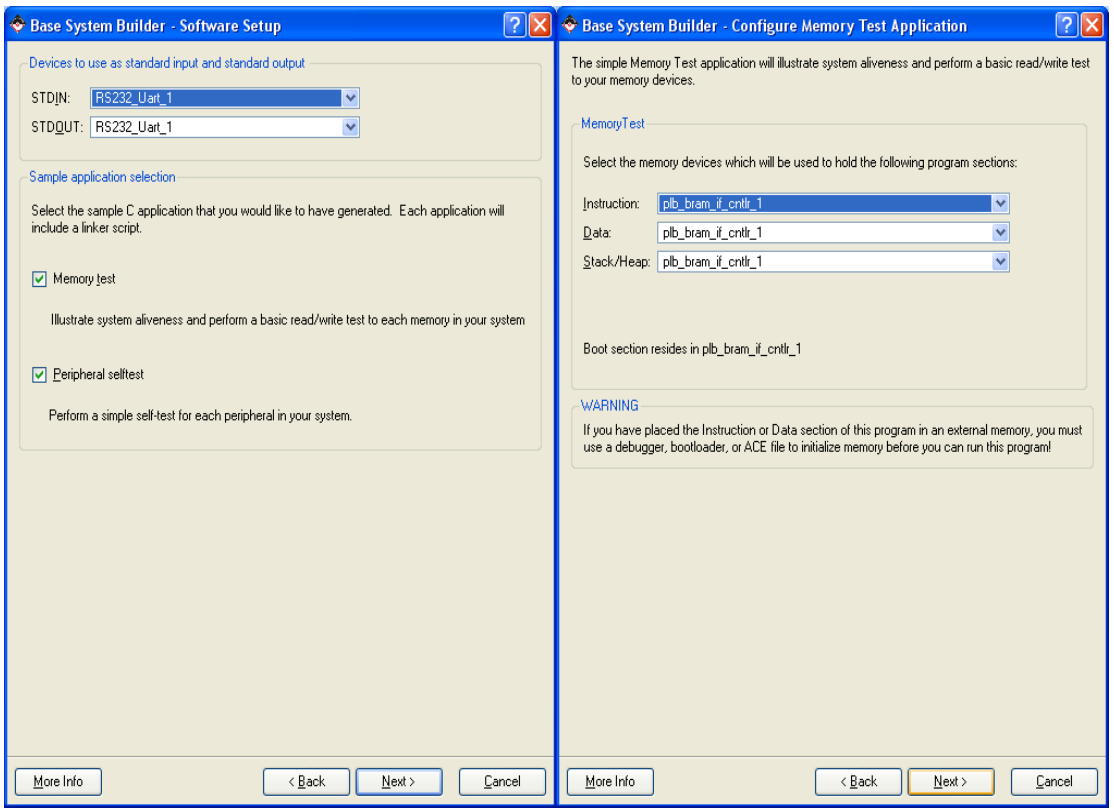
- Añadimos también los switches (DIPSWs_4Bit), los PushButtons_5Bit, la DDR_512MB y los puertos PS2_Ports.



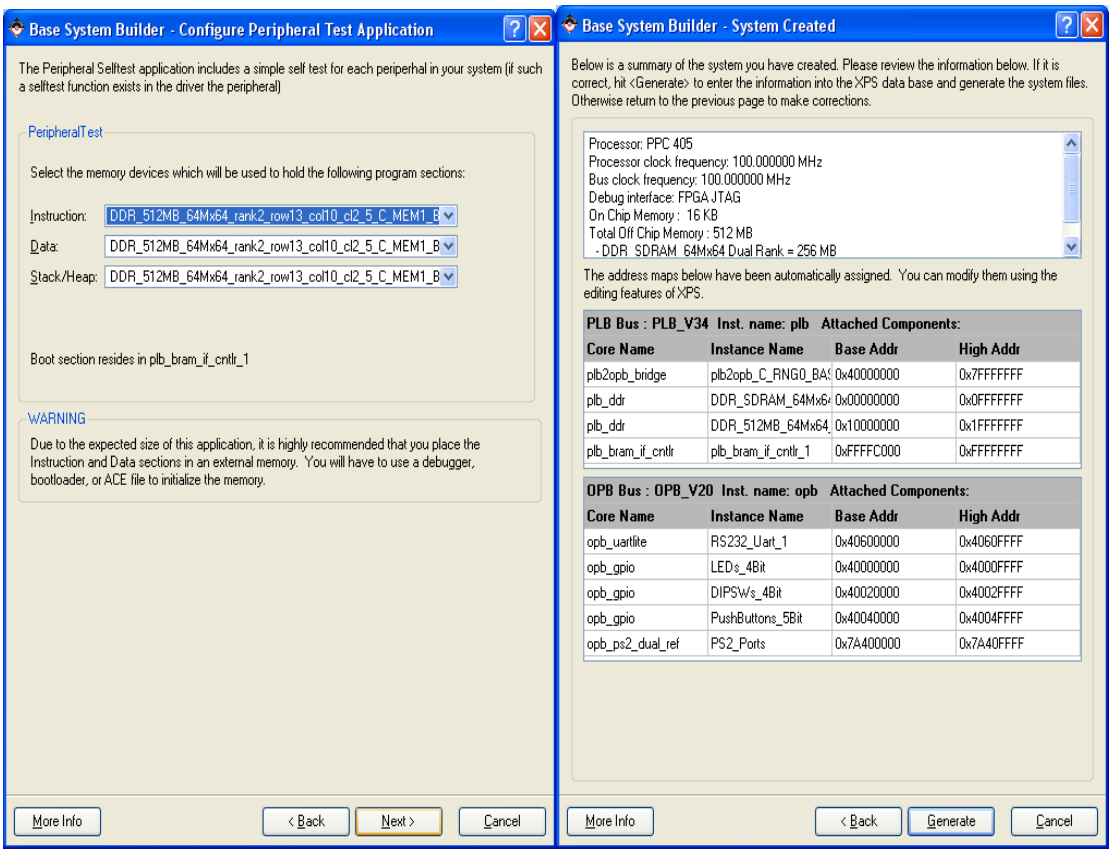
- No añadimos ningún dispositivo de entrada/salida en estas dos pantallas, ya que en el paso siguiente indicaremos que el dispositivo entrada/salida que queremos utilizar es el periférico RS232_Uart1.



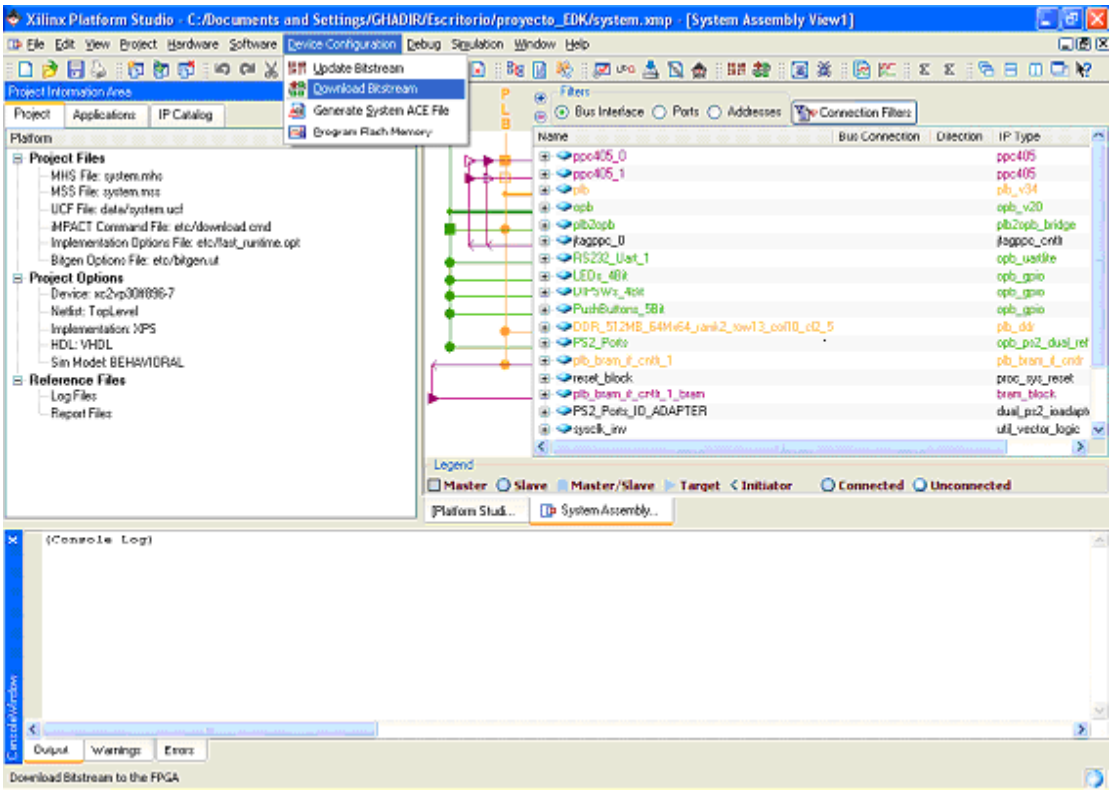
- Se utiliza la UART como entrada y salida estándar. También se da la posibilidad de incluir dos pequeñas aplicaciones de test (una de memoria y otra de dispositivos periféricos). Es recomendable seleccionarlas pues permiten reutilizar código en el que basar la aplicación que se quiere desarrollar, así como para comprobar que los dispositivos funcionan correctamente. En el presente proyecto se ha elegido almacenar el test de memoria en la memoria interna de la FPGA.



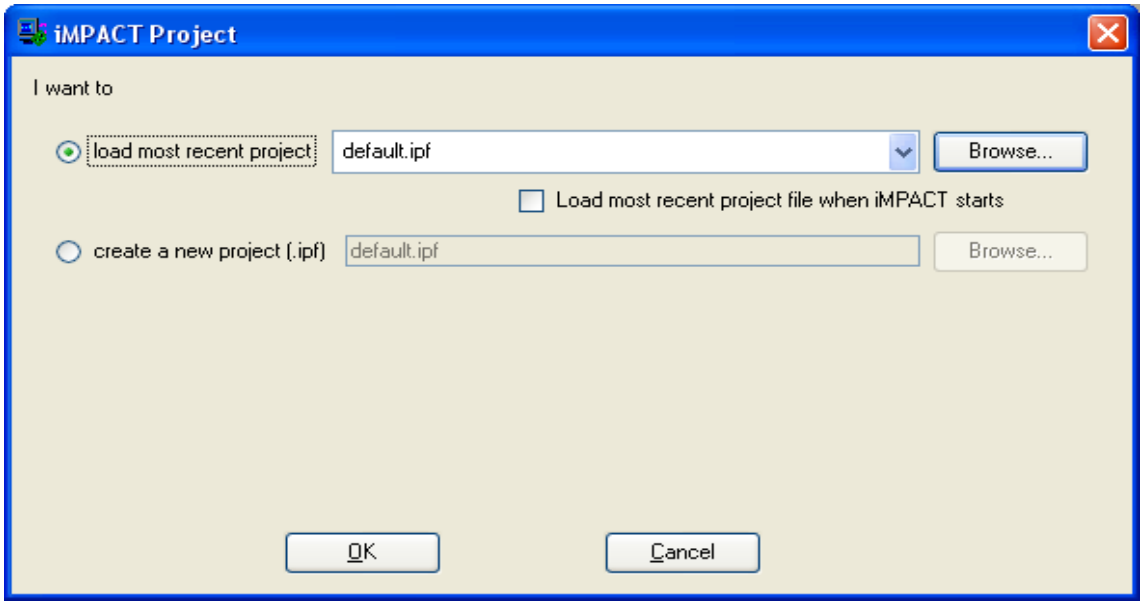
- También se selecciona la DDR para almacenar el test de los dispositivos periféricos. A continuación se muestra un cuadro resumen del mapeo de los dispositivos elegidos en la memoria.



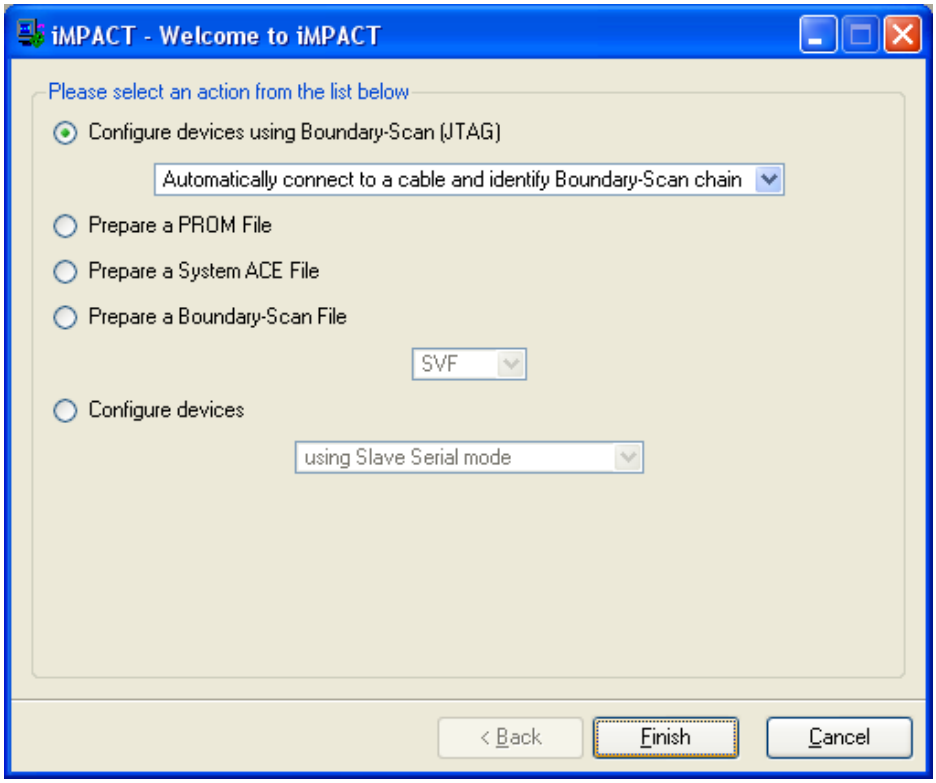
- Una vez se hayan seleccionado todos los dispositivos y los tests se crea el bitstream, que es el fichero de configuración creado a partir de los dispositivos elegidos anteriormente en la herramienta de Xilinx Platform Studio, que contiene la información de la implementación de nuestro diseño en la FPGA.



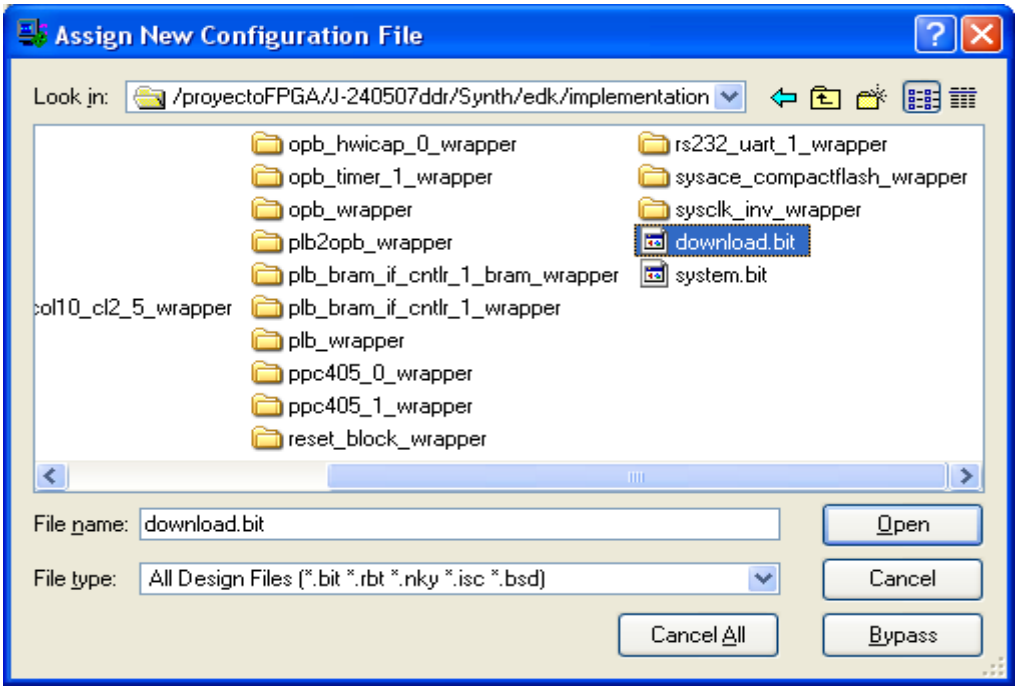
- Para mapear en la FPGA el bitstream se utiliza el programa IMPACT. De esta forma, se informa a la FPGA de aquellos dispositivos que están activos y del código del proyecto. Para ello, abrimos el “Impact” y seleccionamos “create a new Project(.ipf)”, indicando la ruta donde tenemos guardado nuestro .bit



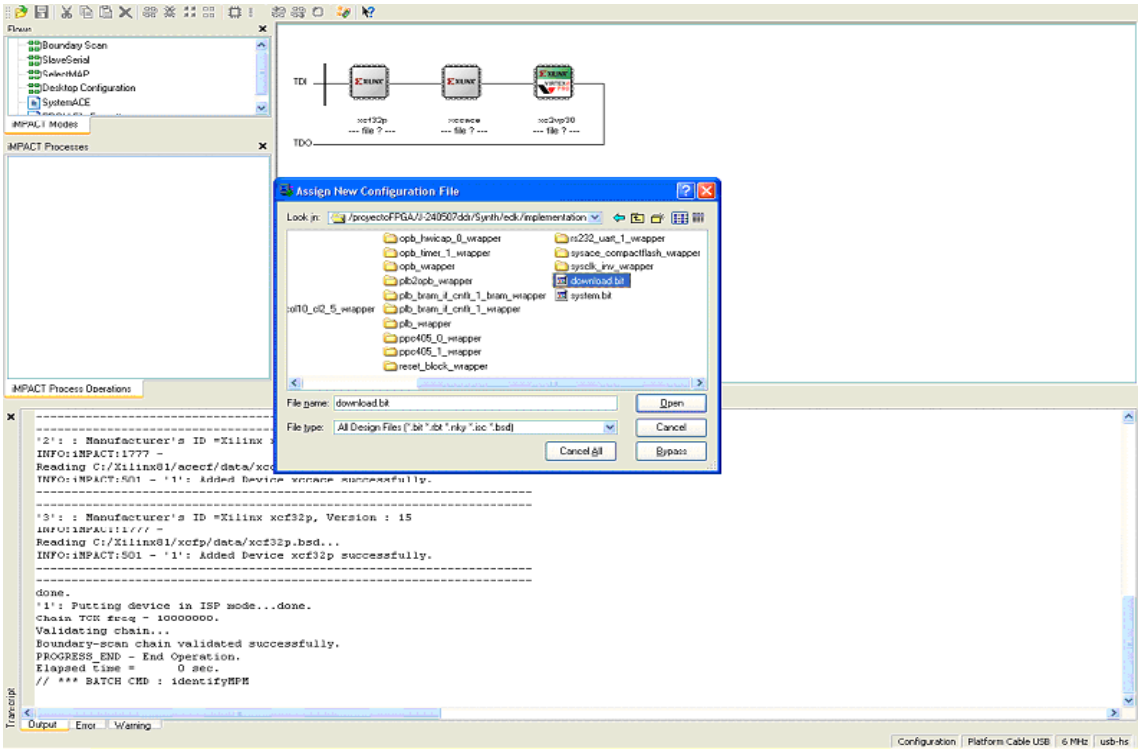
- Se selecciona el modo Boundary-Scan, con lo que se indica a la herramienta que identififique los componentes que hay en la placa por medio del protocolo Boundary-Scan(JTAG), que es el modo que viene seleccionado por defecto.



- Se indica el bitstream que se desea cargar en la FPGA.



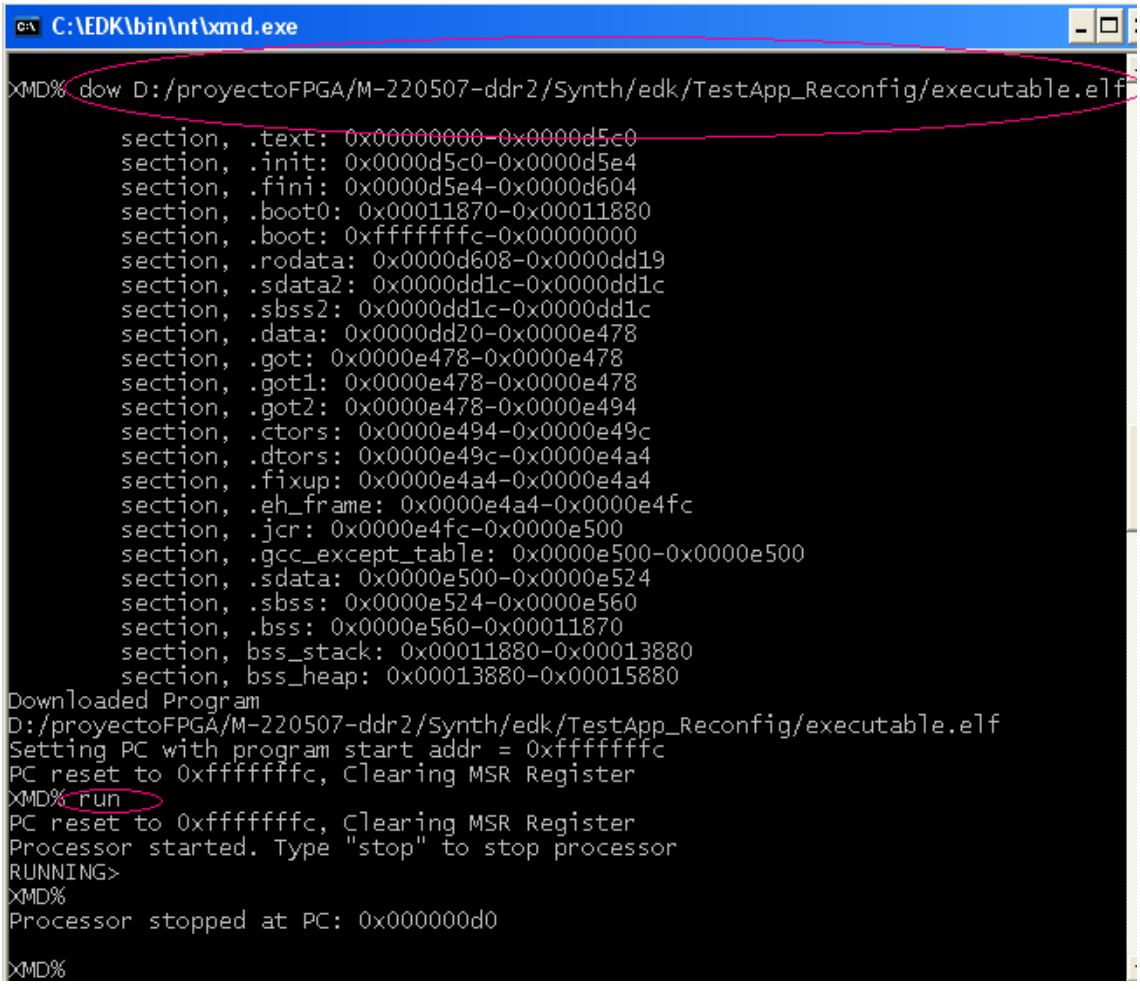
- Se selecciona la FPGA Virtex-II Pro, que es la FPGA en la que se cargará el .bit.



- Una vez que ya se tiene cargado el .bit en la FPGA, faltaría cargar en la placa el archivo executable.elf, que será el fichero ejecutable. Para cargar el ejecutable se utiliza el **XMD,** que descarga el código de la aplicación en la FPGA.

- El acceso al XMD se encuentra en la barra de herramientas del XPS. La herramienta XMD permite elegir el procesador. Una vez seleccionado el Power PC se crea una conexión.
  Si la conexión ha tenido éxito, se procederá a la carga del archivo .elf(el código del proyecto) en la DDR de la FPGA, mediante la instrucción **dow**

- Para terminar se ejecuta el código mediante la instrucción **run.**



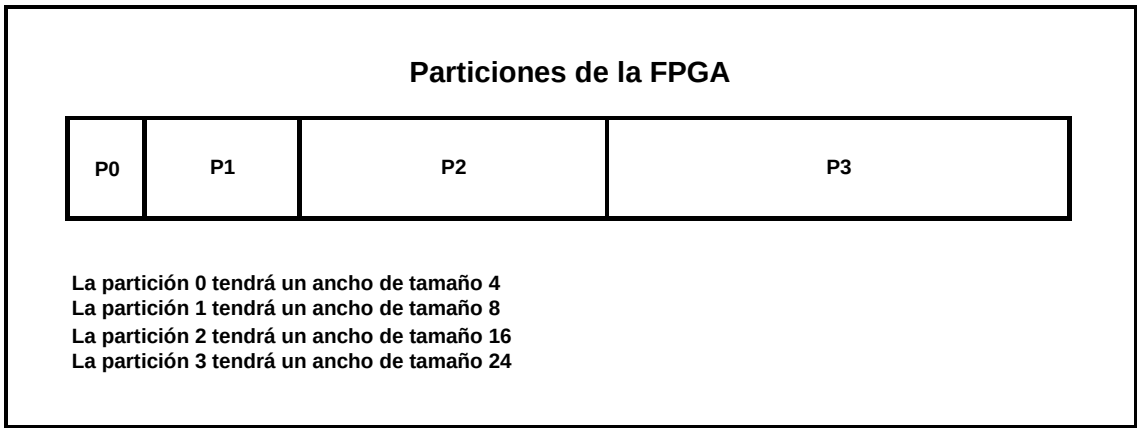
6. Descripción del Algoritmo Utilizado

La implementación del algoritmo planificador de tareas se basa en una serie de criterios que se citan a continuación:

- El algoritmo a desarrollar se centrará en el mantenimiento de la información de las tareas a ejecutar, el control de admisión / planificación de dichas tareas y la ubicación de éstas en una zona concreta de la FPGA.
- Para ello se considerará la FPGA dividida en cuatro particiones de diferentes tamaños caracterizadas por una única dimensión. Todas las particiones tendrán la misma altura pero diferirán en su anchura.

  La altura de la partición coincide con la de un CLB, bloques básicos reconfigurables en los que se divide cada partición.

  Podría verse una partición como una celda cuya altura coincide con la altura de un CLB y cuyo ancho dependerá del número de "columnas" de CLBs que haya en dicha partición.



- Para simular la llegada de las tareas, la información asociada a las tareas será obtenida a través de un fichero de texto, el cual contendrá información detallada para cada una de las tareas:

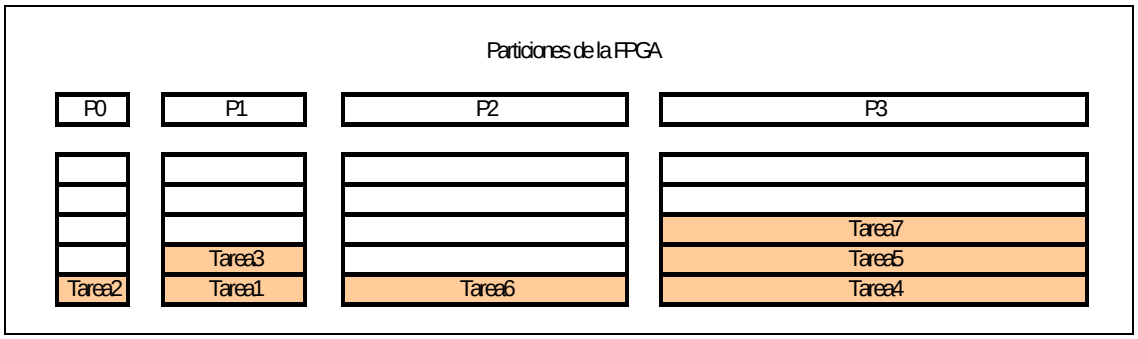
  - Número de la tarea.
  - Tamaño de la tarea, el cual determinará la partición o particiones en las que podría emplazarse.
  - Tiempo o instante de llegada de la tarea a la FPGA
  - Tiempo deadline o tiempo máximo antes del cual la tarea debería estar ejecutada.
  - Tiempo de ejecución asociado a la tarea.
  - Dirección de la primera instrucción compilada a ejecutarse en caso de que la tarea se asigne a la partición 1.
  - Dirección de la primera instrucción compilada a ejecutarse en caso de que la tarea se asigne a la partición 2.

- Dirección de la primera instrucción compilada a ejecutarse en caso de que la tarea se asigne a la partición 3.
- Dirección de la primera instrucción compilada a ejecutarse en caso de que la tarea se asigne a la partición 4.

Es necesario especificar que las direcciones de compilación podrían eliminarse ya que dispondremos de cuatro versiones diferentes para cada tarea, una por cada partición en que dicha tarea pueda asignarse.

- La información de las tareas en el fichero estará ordenada por tiempo de llegada de las tareas.

- Cada una de las particiones de la FPGA dispondrá de una cola de tareas asociada, en la cual se irán guardando las tareas que – una vez planificadas – sean asignadas a dicha partición.

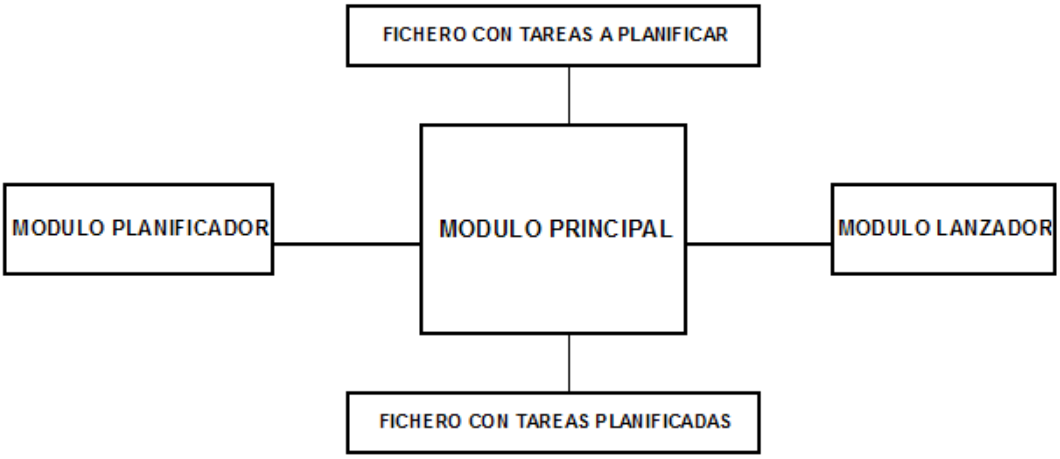


- La asignación de una tarea a una partición concreta de la FPGA dependerá principalmente de dos criterios:

    - El tamaño de la tarea. La tarea sólo podría asignarse a aquellas particiones cuyo ancho de partición sea como mínimo el ancho de la tarea.
    - El tiempo de la tarea. La tarea sólo podría asignarse a una partición si su ejecución en dicha partición termina en un instante de tiempo inferior o igual al de la tarea. La ejecución de una tarea en una partición dependerá de la reconfiguración y ejecución del resto de tareas que se encuentren en la cola asociada a la partición.

- La tarea a planificar se asignará por tanto a la primera partición en la que se cumplan las restricciones establecidas por su tamaño y su deadline.
- En caso de que la tarea no pueda asignarse a ninguna partición será descartada.
- Sólo podrá llegar una tarea por ciclo de reloj y es posible que haya ciclos de reloj en los que no lleguen tareas.
- La aplicación deberá simular un Clock que irá marcando los instantes de tiempo en los que actuará el planificador de tareas.
- Cada partición de la FPGA debe reconfigurarse para poder ejecutar la nueva tarea y el tiempo de reconfiguración es fijo para cada partición y proporcional al tamaño de ésta.
- No podrán reconfigurarse varias particiones de la FPGA simultáneamente pero sí podrán ejecutarse a la vez tareas emplazadas en distintas particiones.

- Será necesario disponer de varios módulos que trabajen conjuntamente:
    - Un módulo **planificador**: que gestione las tareas y decida en función de sus restricciones de tamaño y tiempo máximo de finalización de ejecución (deadline) cuál es la partición de la FPGA en la que serán ejecutadas.
      Esto implica decidir en qué instante de tiempo llegará la tarea a la partición y cuáles serán sus intervalos de tiempos para reconfiguración de la partición y para ejecución de la tarea en dicha partición.
      Este módulo mantendrá la información de las colas.
    - Un módulo **lanzador** que tendrá como función analizar las colas para ubicar las tareas de las colas en la FPGA en el momento adecuado.
      Se encargará por tanto de cargar las tareas en las particiones así como de gestionar su salida una vez terminada la ejecución.
    - Un módulo **principal** que gestione el funcionamiento de los módulos planificador y lanzador.



La ejecución del algoritmo utilizará como entrada un fichero que simulará la llegada de las tareas y su información asociada, y generará un fichero de salida con la planificación de los tiempos de reconfiguración y ejecución de cada tarea, así como la partición asociada.

A continuación se realiza un análisis más exhaustivo de la gestión que realiza cada uno de los módulos de nuestro algoritmo:

Módulo Principal

Este módulo especificará los tipos de datos y gestionará el flujo principal del algoritmo.

Las principales estructuras o tipos de datos requeridos por el algoritmo son:

- Señal de reloj.

  CLK

- Contador de tareas leídas del fichero.

  contador

- Contador de tareas planificadas.

  contador_planificadas

- Contador de tareas lanzadas.

  contador_lanzadas

- Contador de tareas descartadas.

  cont_tareas_descartadas

- Tamaño de las particiones de la FPGA.

  t

- Tiempos de reconfiguración asociados a cada una de las particiones de la FPGA.

  tr

- Indicador boolean que indica si la última línea leída del fichero se corresponde con una tarea ya planificada o no.

  linea_tratada

- Indicador boolean que indica si se ha terminado de leer el fichero de tareas.

  fin_lectura

- Indicador boolean que indica si el algoritmo ha terminado su ejecución total.

  FIN

- Mantenimiento de los intervalos de reconfiguración para cada una de las tareas planificadas.

  v_reconf_fpga

- Información de la tarea leída del fichero.

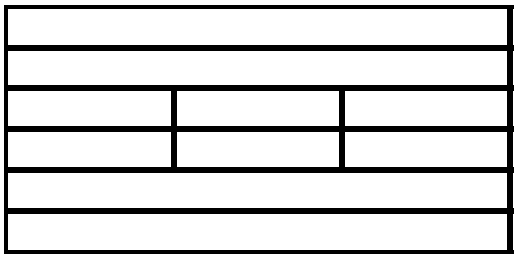
Tarea	Tamaño	Tiempo llegada	Delay	Dir compilación P0	Dir compilación P1	Dir compilación P2	Dir compilación P3

info_tarea

- Información de la tarea que se introducirá en la cola.

Tarea	Tamaño	Tiempo llegada	Tiempo	Deadline	Dir.com	Inicio Reconfiguración

- Información de cada cola.



donde:

- contador es el número total de tareas que hay en la cola.
- tiempo restante es el instante de tiempo en que terminará la reconfiguración y ejecución de todas las tareas de la cola. Puede verse como el instante en el cual una nueva tarea podría reconfigurarse en la partición.
- tareas es un vector de tareas e información asociada a éstas.
- tiempos de ejecución es un vector que guarda para cada tarea la tupla de valores que indican el inicio y el fin de reconfiguración de la tarea en la partición de la FPGA asociada a la cola.
- contador ejecutadas indica el número de tareas cuya planificación ha sido especificada.
- contador lanzadas indica el número de tareas cuya ejecución ha sido especificada.

- Colas asociadas a las particiones de la FPGA.

Cola 0	Cola 1	Cola 2	Cola 3

colas

A continuación se muestra información más detallada sobre cada uno de los módulos.

Módulo Principal

Este módulo realizará todas las inicializaciones necesarias y gestionará la señal del CLK y las llamadas a los módulos Planificador y Lanzador.

El pseudo código de este módulo se muestra a continuación:

```
main(){

inicializar_fichero_salida();
inicializar_fichero_entrada();
inicializar_colas();
inicializar_tamaño_particiones();
inicializar_vector_reconfiguraciones();
inicializar_contadores();
inicializar_clock();

FIN=false
fin_lectura=false;
linea_tratada=trae;

Mientras ( FIN = false ) hacer
{
        esperar_llegada_señal_temporizador();
        CLK = CLK +1;
        ejecutar_planificador();
        ejecutar_lanzador();
}

cerrar_fichero_entrada();
cerrar_fichero_salida();
```

Módulo Planificador

Este módulo tiene como objetivo buscar una partición de la FPGA en la que la tarea pueda ejecutarse cumpliendo las restricciones de tamaño y de la tarea. La tarea será guardada en la cola asociada a la partición elegida.

El pseudo código de este módulo se muestra a continuación:

```
ejecutar_planificador(){
        si ( (fin_lectura==false)&&(linea_tratada==trae) {leer_linea();}
        si (CLK=tiempo_llegada_de_la_tarea){buscar_particion_a_tarea();}
}

buscar_particion_a_tarea(){
        si encuentro_particion_para_tarea() {
                        guardar_tarea_en_cola();
                        actualizar_informacion_de_la_cola();
                        actualizar_vector_de_tiempos_de_reconfiguraciones();
        }
        sino {descartar_tarea();}
}
```

Módulo Lanzador

Este módulo tiene como objetivo lanzar una tarea.
Para ello escoge la primera tarea no lanzada de cada cola y se queda con aquella que tenga menor tiempo de inicio de reconfiguración.
Si ese tiempo de inicio de reconfiguración coincide con el CLK actual se lanza la tarea.
Por último comprueba si se debe poner FIN a trae, ayudándose del campo tiempo_restante incluido en las colas.

El pseudo código de este módulo se muestra a continuación:

```
ejecutar_lanzador(){

        si hay_tareas_planificadas_sin_lanzar() {

                coger_la_primera_tarea_no_lanzada_de_cada_cola();
                elegir_de_entre_ellas_la_de_menor_tiempo_de_reconfiguracion();
                comprobar_si_CLK_coincide_con_ese_tiempo_de_reconfiguracion();

                        si coinciden(){
                                lanzar_tarea();
                        }

                comprobar_si_FIN_es_true();

        }

}
```

7. Resultados y Conclusiones

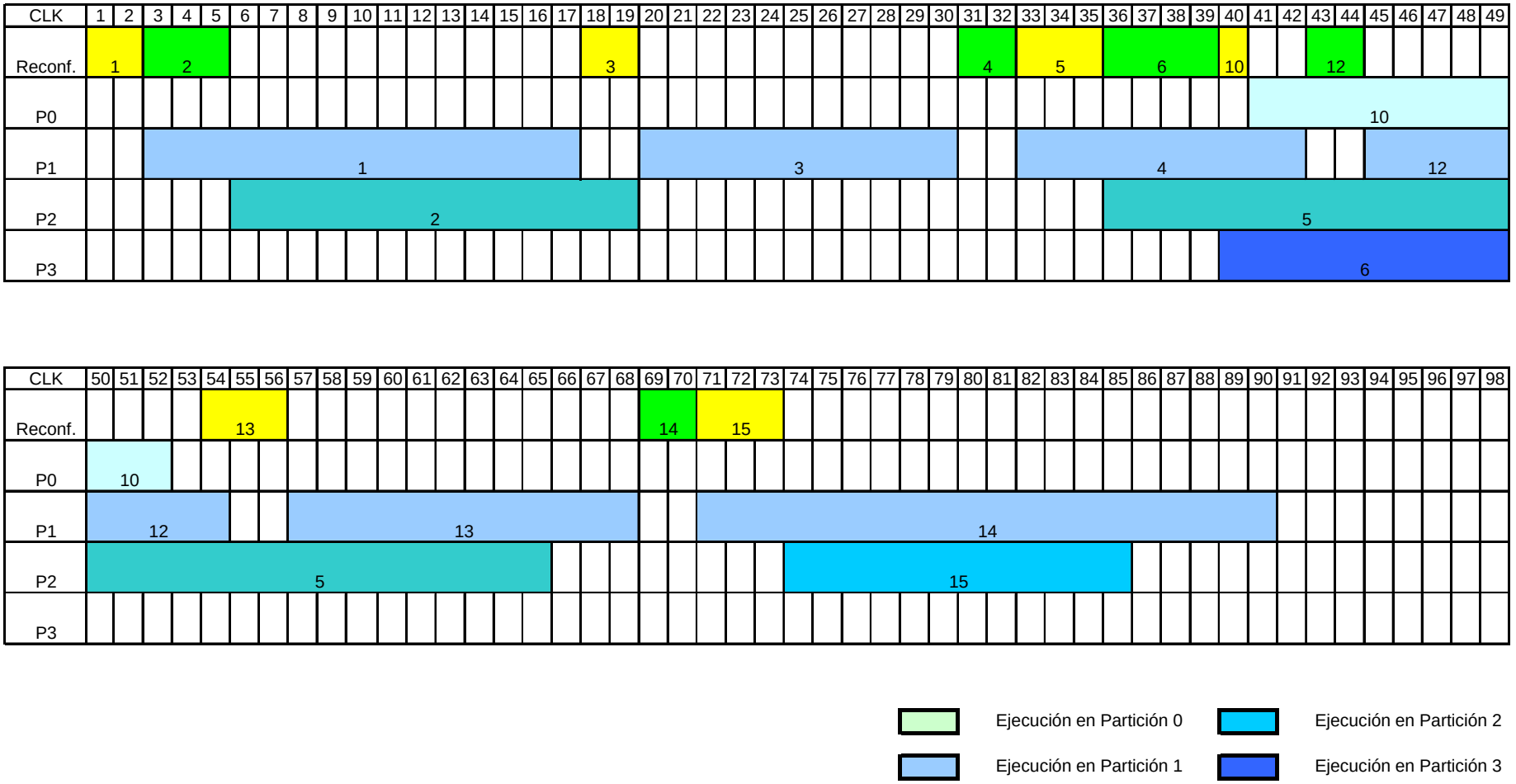
A continuación se incluyen ejemplos de funcionamiento del algoritmo, que permitirán realizar comparativas y extraer conclusiones.

Ejemplo 1

Lote de tareas:

Tarea	Tamaño	Tiempo de Llegada	Tiempo de Ejecución	Deadline
1	5	1	15	100
2	10	2	14	30
3	5	3	11	80
4	5	4	10	66
5	10	5	30	70
6	16	6	10	55
7	7	7	20	60
8	3	8	10	50
9	2	9	18	41
10	3	10	12	62
11	9	11	20	60
12	8	12	10	60
13	7	13	12	100
14	8	14	20	180
15	10	15	12	100

La ejecución del lote de tareas termina en el instante de reloj 90 y se descartan un total de 4 tareas.
A continuación se muestra la traza de ejecución mostrando los intervalos de reconfiguración de las tareas así como los ciclos de reloj en los que se está ejecutando alguna tarea en cada una de las particiones.

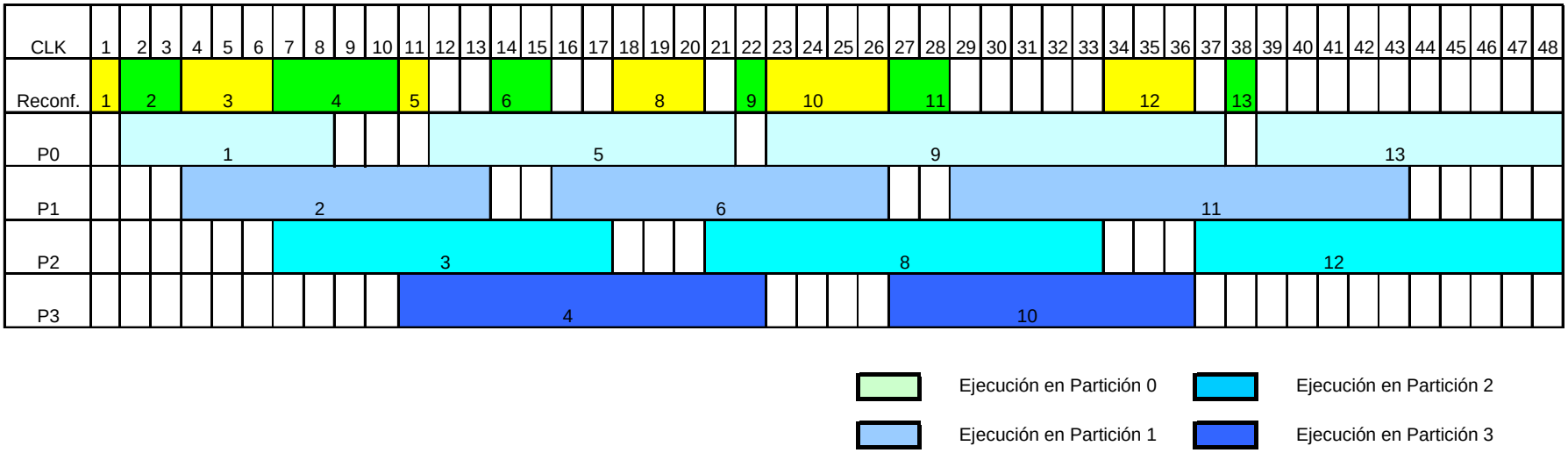

CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Reconf.
1
2
3
4
5
6
10
12
P0
10
P1
1
3
4
12
P2
2
5
P3
6
CLK
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Reconf.
13
14
15
P0
10
P1
12
13
14
P2
5
15
P3
Ejecución en Partición 0
Ejecución en Partición 1
Ejecución en Partición 2
Ejecución en Partición 3

Ejemplo 2

Lote de tareas:

Tarea	Tamaño	Tiempo de Llegada	Tiempo de Ejecución	Deadline
1	3	1	7	11
2	6	2	10	15
3	10	3	11	20
4	20	4	12	25
5	2	5	10	25
6	7	6	11	30
7	26	7	20	30
8	15	10	13	35
9	3	12	15	50
10	23	14	10	70
11	6	16	15	50
12	15	20	12	70
13	3	21	10	80
14	30	22	12	70
15	35	23	10	80

La ejecución del lote de tareas termina en el instante de reloj 49 y se descartan un total de 3 tareas.
A continuación se muestra la traza de ejecución mostrando los intervalos de reconfiguración de las tareas así como los ciclos de reloj en los que se está ejecutando alguna tarea en cada una de las particiones.


CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
Reconf.
1 2 3 4 5 6 8 9 10 11 12 13
P0
1 5 9 13
P1
2 6 11
P2
3 8 12
P3
4 10
Ejecución en Partición 0
Ejecución en Partición 1
Ejecución en Partición 2
Ejecución en Partición 3

La siguiente tabla muestra los resultados obtenidos:

CRITERIO	LOTE 1	LOTE 2
Número de tareas en el lote	15	15
Número de tareas descartadas del lote	4	3
Tiempo total de ejecución	90	48
Porcentaje de tiempo en reconfiguración	30%	50%
Porcentaje de tiempo en ejecución	93%	98%

En la tabla anterior puede observarse que la mayor parte del tiempo total la FPGA se encuentra ejecutando. En el primer lote un 93% frente al 98% para el segundo lote.

Es aquí donde se muestran las ventajas de la reconfiguración parcial dinámica, la cual nos permite ejecutar en distintas particiones simultáneamente , e incluso realizar una reconfiguración y varias ejecuciones a la vez, siempre que en una misma partición no se reconfigure y ejecute al mismo tiempo.

El segundo lote muestra un conjunto de tareas más “ideal”, en el que las tareas van llegando a particiones sucesivas, lo que hace que apenas haya tiempo de espera entre la llegada de la tarea y el inicio de su reconfiguración.

Sin embargo también es posible encontrar lotes de tareas cuyas características hacen que la FPGA tenga grandes intervalos de tiempo en los que no se puede llevar a cabo reconfiguración. Véase el siguiente lote de tareas.

Ejemplo 3

Lote de tareas:

Tarea	Tamaño	Tiempo de Llegada	Tiempo de Ejecución	Deadline
1	100	1	2	4
2	10	2	3	8
3	5	3	4	11
4	5	4	3	15
5	10	5	2	18
6	15	6	3	21
7	7	7	5	25
8	3	8	25	45
9	2	9	20	41
10	3	10	20	62
11	15	11	10	60
12	20	12	10	60
13	15	13	5	63
14	80	14	10	200
15	20	15	20	100

La ejecución del lote de tareas termina en el instante de reloj 82 y se descartan un total de 3 tareas.
A continuación se muestra la traza de ejecución mostrando los intervalos de reconfiguración de las tareas así como los ciclos de reloj en los que se está ejecutando alguna tarea en cada una de las particiones.

En la traza puede observarse cómo la tarea 10 debe retrasar su reconfiguración hasta que finalice la ejecución de la tarea 9, ya que ambas tareas van a la misma partición.
El retraso producido en la tarea 10 provoca a su vez retrasos en la reconfiguración de las tareas siguientes (11, 12, 13 y 15) ya que éstas no podrán reconfigurarse hasta que no se haya reconfigurado la tarea 10.
Esto se debe a que la política de reconfiguración se basa en la numeración de las tareas, de modo que la tarea i siempre se reconfigurará antes que la tarea i+1.
Sería por tanto interesante modificar esta política de reconfiguración de modo que el orden de reconfiguración de las tareas no tenga por qué ser idéntico al orden de llegada de las tareas a la FPGA. Esto evitaría los huecos vacíos de reconfiguración y permitiría que la ejecución de todas las tareas finalizara mucho antes.

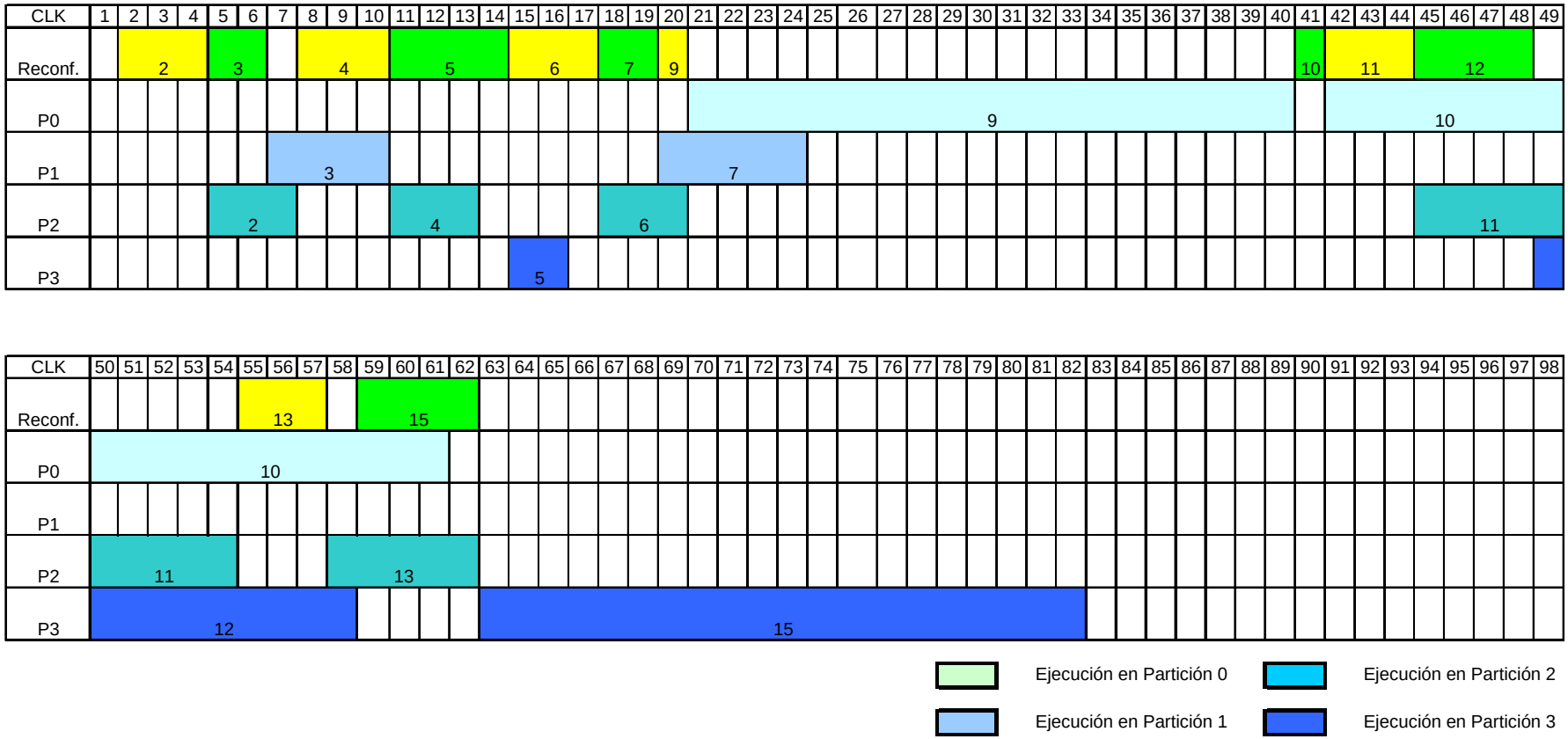
Observaciones:

- Las pruebas realizadas permiten comprobar que el punto débil del algoritmo utilizado es no permitir que las tareas se reconfiguren en un orden distinto al de llegada.

- Sería por tanto interesante modificar la política de reconfiguración.

- Esto evitaría los huecos vacíos de reconfiguración y permitiría que la ejecución de todas las tareas finalizara mucho antes.

- Una eficiente política de reconfiguración unida a los beneficios de la multiejecución HW y SW abren un amplio camino de investigación .


CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
Reconf.
2 3 4 5 6 7 9 10 11 12
P0
9 10
P1
3 7
P2
2 4 6 11
P3
5
CLK
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
Reconf.
13 15
P0
10
P1
P2
11 13
P3
12 15
Ejecución en Partición 0
Ejecución en Partición 1
Ejecución en Partición 2
Ejecución en Partición 3

Para finalizar se muestra un ejemplo del fichero generado por el algoritmo para simular la reconfiguración y ejecución de las tareas en la FPGA.

El lote de tareas utilizado será el siguiente:

Tarea	Tamaño	Tiempo de Llegada	Tiempo de Ejecución	Deadline
1	100	1	2	4
2	10	2	3	8
3	5	3	4	11
4	5	6	3	15
5	10	7	2	18

La ejecución del algoritmo (planificador / lanzador) genera un archivo que incluye la traza de sucesos en la ejecución. Se muestra a continuación:

```
      Se van a planificar y ejecutar un total de 5 tareas


************** INSTANTE DE CLOCK 1 **************

************************PLANIFICADOR*******************************

-------------------LLEGA NUEVA TAREA-----------------------------

El instante de llegada es: 1
El tiempo deadline es: 4
El tiempo de ejecución es: 2.000000
El tamaño es: 100

-------------------INTENTO ASIGNARLE UNA COLA---------------------

Atención!! Imposible asignar en cola ** 0 ** por TAMAÑO

Atención!! Imposible asignar en cola ** 1 ** por TAMAÑO

Atención!! Imposible asignar en cola ** 2 ** por TAMAÑO

Atención!! Imposible asignar en cola ** 3 ** por TAMAÑO

La tarea numero 1 ha sido descartada

El total de tareas descartadas es: 1


*************************LANZADOR********************************

No hay tareas para lanzar


************** INSTANTE DE CLOCK 2 **************

************************PLANIFICADOR*******************************

-------------------LLEGA NUEVA TAREA-----------------------------

El instante de llegada es: 2
El tiempo deadline es: 8
El tiempo de ejecución es: 3.000000
El tamaño es: 10

-------------------INTENTO ASIGNARLE UNA COLA---------------------

Atención!! Imposible asignar en cola ** 0 ** por TAMAÑO

Atención!! Imposible asignar en cola ** 1 ** por TAMAÑO
```

```
Cumple condiciones de tamaño y deadline de la cola ** 2 **

El tiempo de comienzo de reconfiguración es ---> 2.000000

El intervalo de reconfiguración es ( 2.000000 , 5.000000 )

El intervalo de ejecución es ( 5.000000 , 8.000000 )

El tiempo de espera de la cola queda en: 8.000000


      La cola 0 tiene 0 tareas y su tiempo de espera es 0.000000

      La cola 1 tiene 0 tareas y su tiempo de espera es 0.000000

      La cola 2 tiene 1 tareas y su tiempo de espera es 8.000000

      La cola 3 tiene 0 tareas y su tiempo de espera es 0.000000


************************LANZADOR*******************************

La cola elegida para sacar tarea es la numero 2

LA TAREA Y SU INSTANTE DE RECONFIGURACION SON:
  TAREA: 2.000000
  TIEMPO DE RECONFIGURACION: 2


*************** INSTANTE DE CLOCK 3 *************

************************PLANIFICADOR*******************************

-------------------LLEGA NUEVA TAREA-----------------------------

El instante de llegada es: 3
El tiempo deadline es: 11
El tiempo de ejecución es: 4.000000
El tamaño es: 5

-------------------INTENTO ASIGNARLE UNA COLA---------------------

Atención!! Imposible asignar en cola ** 0 ** por TAMANO

Cumple condiciones de tamaño y deadline de la cola ** 1 **

El tiempo de comienzo de reconfiguración es ---> 5.000000

El intervalo de reconfiguración es ( 5.000000 , 7.000000 )

El intervalo de ejecución es ( 7.000000 , 11.000000 )

El tiempo de espera de la cola queda en: 11.000000
```

```
      La cola 0 tiene 0 tareas y su tiempo de espera es 0.000000

      La cola 1 tiene 1 tareas y su tiempo de espera es 11.000000

      La cola 2 tiene 1 tareas y su tiempo de espera es 8.000000

      La cola 3 tiene 0 tareas y su tiempo de espera es 0.000000

**************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 1

El tiempo de reconfiguración de la tarea elegida no coincide con este
instante de reloj:

    TIEMPO DE RECONFIGURACION = 5 y CLOCK = 3


********** INSTANTE DE CLOCK 4 **********

************************PLANIFICADOR*******************************

No ha llegado tarea en CLK = 4

**************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 1

El tiempo de reconfiguración de la tarea elegida no coincide con este
instante de reloj:

    TIEMPO DE RECONFIGURACION = 5 y CLOCK = 4


********** INSTANTE DE CLOCK 5 **********

************************PLANIFICADOR*******************************

No ha llegado tarea en CLK = 5

**************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 1

LA TAREA Y SU INSTANTE DE RECONFIGURACION SON:
  TAREA: 3.000000
  TIEMPO DE RECONFIGURACION: 5


********** INSTANTE DE CLOCK 6 **********

************************PLANIFICADOR*******************************

------------------LLEGA NUEVA TAREA-----------------------------

El instante de llegada es: 6
El tiempo deadline es: 15
El tiempo de ejecución es: 3.000000
El tamaño es: 5
```

```
------------------INTENTO ASIGNARLE UNA COLA--------------------

Atención!! Imposible asignar en cola ** 0 ** por TAMANO

Atención!! Imposible asignar en cola ** 1 ** por DEADLINE:

La tarea acabaría en el instante 16.000000 y su deadline es
15

Cumple condiciones de tamaño y deadline de la cola ** 2 **

El tiempo de comienzo de reconfiguración es ---> 8.000000

El intervalo de reconfiguración es ( 8.000000 , 11.000000 )

El intervalo de ejecución es ( 11.000000 , 14.000000 )

El tiempo de espera de la cola queda en: 14.000000


      La cola 0 tiene 0 tareas y su tiempo de espera es 0.000000

      La cola 1 tiene 1 tareas y su tiempo de espera es 11.000000

      La cola 2 tiene 2 tareas y su tiempo de espera es 14.000000

      La cola 3 tiene 0 tareas y su tiempo de espera es 0.000000


************************LANZADOR*******************************

La cola elegida para sacar tarea es la numero 2

El tiempo de reconfiguración de la tarea elegida no coincide con este
instante de reloj:

    TIEMPO DE RECONFIGURACION = 8 y CLOCK = 6


********** INSTANTE DE CLOCK 7 **********

************************PLANIFICADOR*******************************

------------------LLEGA NUEVA TAREA----------------------------


El instante de llegada es: 7
El tiempo deadline es: 18
El tiempo de ejecución es: 2.000000
El tamaño es: 10

------------------INTENTO ASIGNARLE UNA COLA--------------------

Atención!! Imposible asignar en cola ** 0 ** por TAMANO

Atención!! Imposible asignar en cola ** 1 ** por TAMANO

Atención!! Imposible asignar en cola ** 2 ** por DEADLINE:

La tarea acabaría en el instante 19.000000 y su deadline es 18
```

```
Cumple condiciones de tamaño y deadline de la cola ** 3 **

El tiempo de comienzo de reconfiguración es ---> 11.000000

El intervalo de reconfiguración es ( 11.000000 , 15.000000 )

El intervalo de ejecución es ( 15.000000 , 17.000000 )

El tiempo de espera de la cola queda en: 17.000000


      La cola 0 tiene 0 tareas y su tiempo de espera es 0.000000

      La cola 1 tiene 1 tareas y su tiempo de espera es 11.000000

      La cola 2 tiene 2 tareas y su tiempo de espera es 14.000000

      La cola 3 tiene 1 tareas y su tiempo de espera es 17.000000


**************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 2

El tiempo de reconfiguración de la tarea elegida no coincide con este
instante de reloj:

    TIEMPO DE RECONFIGURACION = 8 y CLOCK = 7


********** INSTANTE DE CLOCK 8 **********

**************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 2

LA TAREA Y SU INSTANTE DE RECONFIGURACION SON:
  TAREA: 4.000000
  TIEMPO DE RECONFIGURACION: 8


********** INSTANTE DE CLOCK 9 **********

**************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 3

El tiempo de reconfiguración de la tarea elegida no coincide con este
instante de reloj:

    TIEMPO DE RECONFIGURACION = 11 y CLOCK = 9
```

```
********** INSTANTE DE CLOCK 10 **********

************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 3

El tiempo de reconfiguración de la tarea elegida no coincide con este
instante de reloj:

    TIEMPO DE RECONFIGURACION = 11 y CLOCK = 10



********** INSTANTE DE CLOCK 11 **********

************************LANZADOR*********************************

La cola elegida para sacar tarea es la numero 3

LA TAREA Y SU INSTANTE DE RECONFIGURACION SON:
  TAREA: 5.000000
  TIEMPO DE RECONFIGURACION: 11
Se ha finalizado la planificación

La ejecución de todas las tareas termina en el ciclo de reloj 17
```

7. Conclusiones Generales

El desarrollo del presente proyecto ha permitido comprobar la potencia de la Reconfiguración Parcial Dinámica en sistemas hardware en los que se ejecuta multitarea.

Sin embargo actualmente los tiempos de reconfiguración de las FPGAs son muy costosos en tiempo, lo que limita su aplicación.
La disminución de estos tiempos de reconfiguración crea una interesante perspectiva sobre la multitarea HW y SW.

8. Bibliografía

- http://www.xilinx.com/products/silicon_solutions/fpgas/

- http://h2non.wordpress.com/2007/04/03/

- http://es.wikipedia.org/wiki

- Sara Román, Hortensia Mecha, Daniel Mozos, Julio Septien. Artículo “Partition Based Dynamic 2D HW Multitasking Management”

- *Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, CSIC.* Artículo “**DESARROLLO DE MÓDULOS-IP DE CONTROLADORES DIFUSOS PARA EL DISEÑO DE SISTEMAS EMPOTRADOS SOBRE FPGAS”**